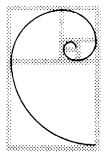


SYBASE®

System 10

System & Database Administration

Volume 1

Student Guide

58161-01-0210-01

Notice:

The information contained in this document is subject to change without notice.

Sybase, Inc. MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Sybase, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Sybase, Inc. assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Sybase Inc.

Course Modules & Appendices

Table of Contents

1 Overview

2 SQL Server Operating Environment

3 Allocating Resources

4 Creating Databases & Segments

5 Controlling Access

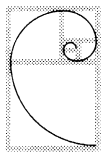
6 Configuring SQL Server

7 Transaction Management

8 Backing Up Databases & Logs

9 Monitoring & Troubleshooting SQL Server

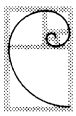
13 Other Server Administration Topics (Optional)



SYBASE®

Module 1

Overview



SYBASE

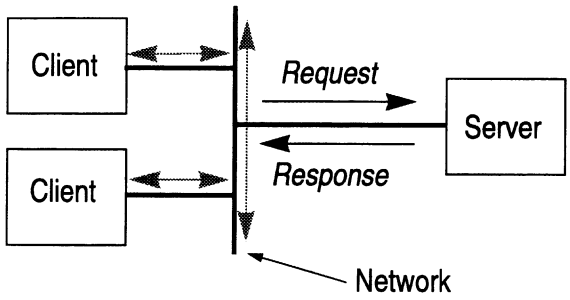
Overview

Objectives

- Describe features of Client/Server architecture
- Describe features of SQL Server
- Describe responsibilities of the SQL Server Administrator
- Describe responsibilities of users with SA, SSO and OPER roles
- Describe system tables in every database and in the *master* database
- Use isql and SA Companion to access SQL Server
- Write script files, and run them against SQL Server

SYBASE

What is Client/Server Architecture?



- An architecture with two types of components that communicate across a network
  - A *client* is a program that submits requests
  - A *server* is a program that responds to these requests

Clients

Clients send requests and handles responses. They might:

- display and manage application work environment and user interface
- perform data validation, display reports, and represent data graphically

Servers

Servers process requests and possibly return results, such as:

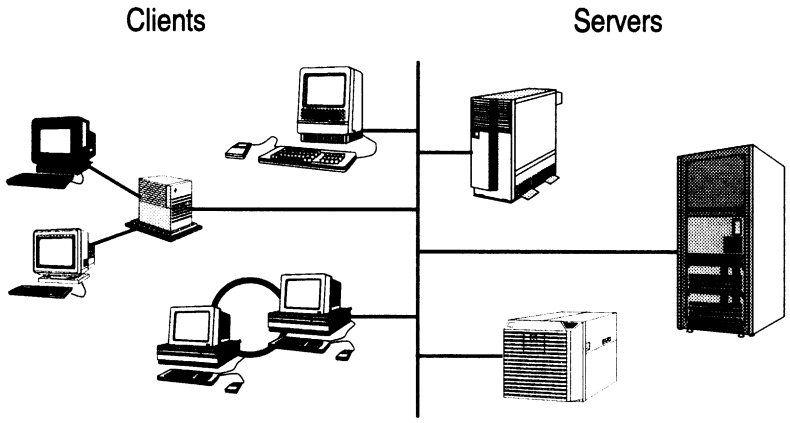
- execute SQL commands sent from a client
- take some control action, such as lowering room temperature

Network

Provides connection between clients and servers, but how network is used is transparent to the clients and servers on it

Hardware Interoperability

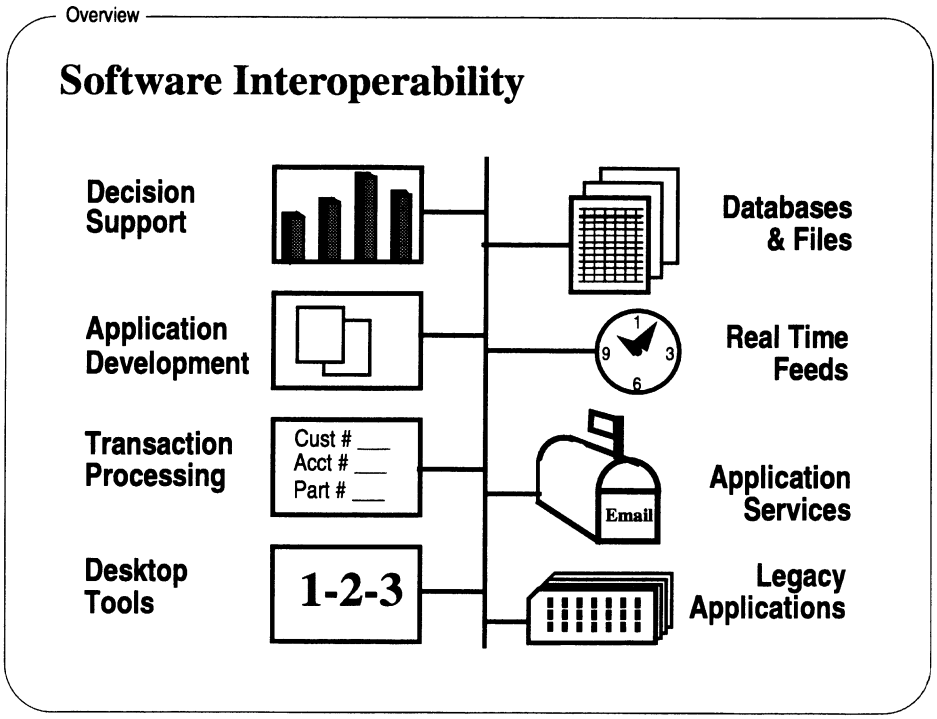
- The Client/Server model allows clients and servers on different kinds of hardware to talk to each other across a network



Hardware

For both client and server, you can choose the hardware that:

- meets your price and performance needs,
- takes advantage of the network technology,
- offers the best graphical interface for the user.



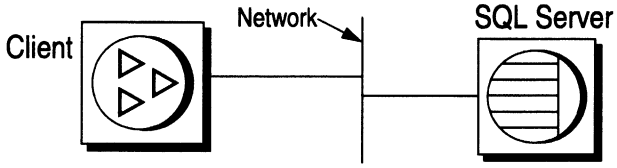
Software interoperability

The Client/Server model allows you to integrate any client to any server, any application to any data source. With Sybase, as we will see in subsequent foils, this is done via Open Client and Open Server.

Overview

Optimal Use of Clients & Data Servers

- Client/Server architecture allows each part of the system to be built optimally



- User-friendly
- Individual orientation
- Low development cost
- Low hardware/software costs

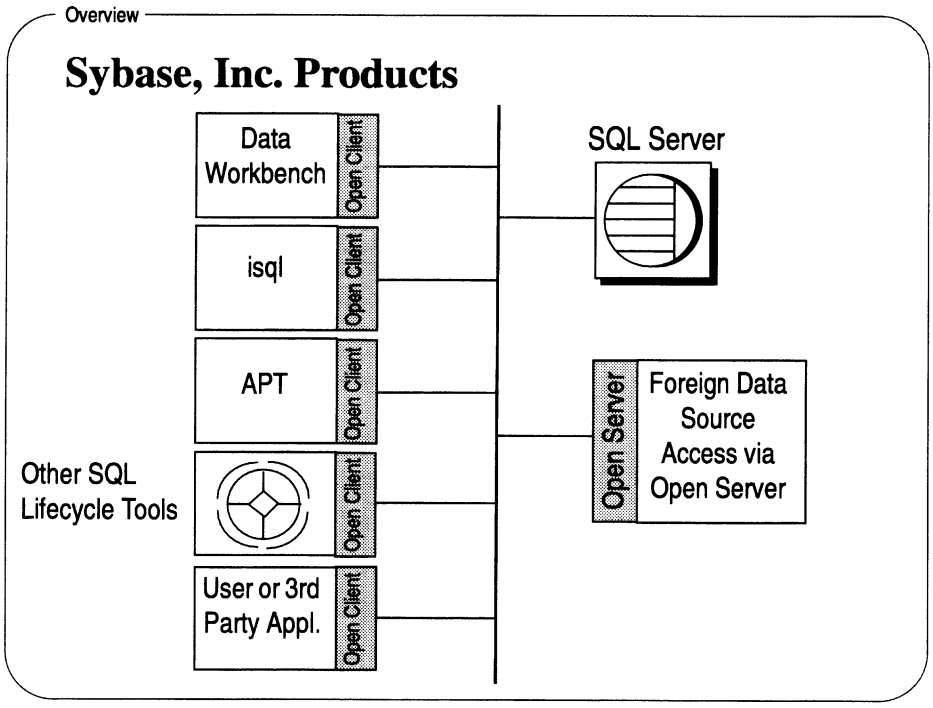
- Data integrity
- Security
- Concurrency
- Availability
- Performance
- Centralized data management

clients

In the Client/Server paradigm, client programs can be built to offer the best interface for the user without having to program in data integrity, security, etc. on the client side.

servers

In this paradigm, server programs, like SQL Server, can be built for data integrity, performance, availability, etc.



Sybase products

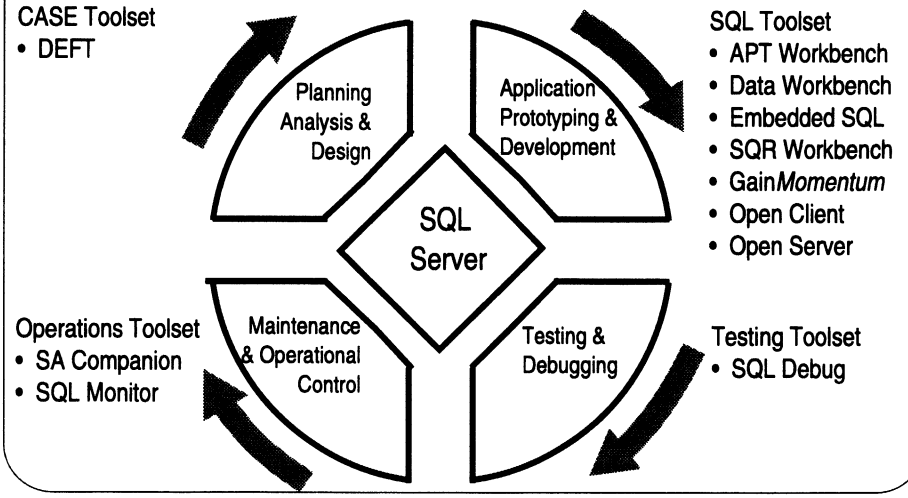
SQL Server manages data and compiles and executes queries. Sybase clients or clients written by users or third party vendors using Open Client (OC) access the server.

Clients can also access other data sources or applications using Open Server applications.

Sybase's Lifecycle Tools provide a complete application development toolset.

SYBASE SQL Lifecycle Tools

- Sybase provides tools for each part of the SQL development lifecycle



Partial list of products

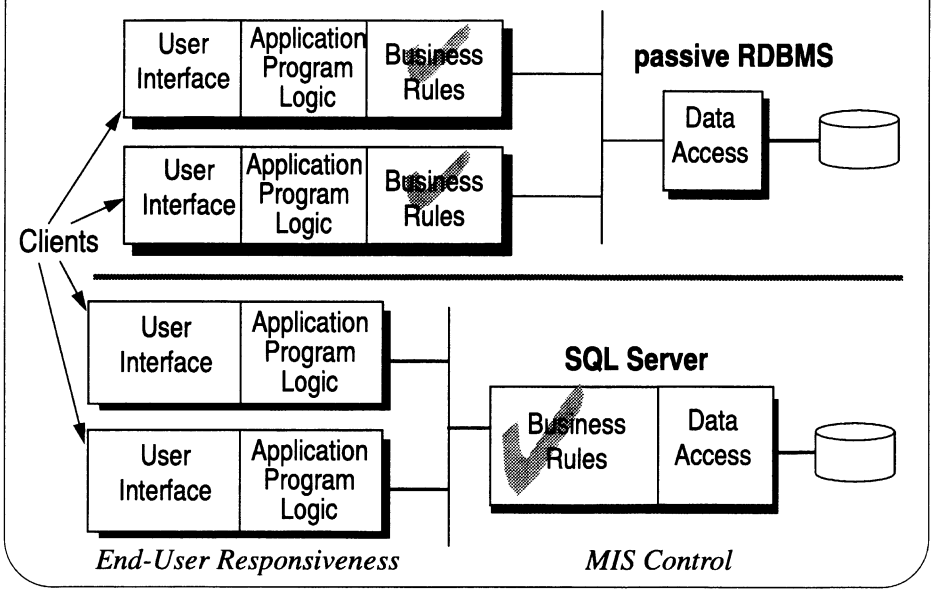
Sybase clients:

- Data Workbench
- isql
- APT & APT/Motif
- SQR Workbench
- Deft
- SQL Debug
- Gain*Momentum*
- Open Client
- Embedded SQL
- SA Companion
- SQL Monitor

Sybase servers:

- SQL Server
- Open Server
- Replication Server
- OmniSQL Gateway

Client/Server Database Application Models



"Passive" RDBMS

Each application has to enforce data integrity and business rules on its own. Disadvantages:

- Application maintenance is difficult.
  If a business rule is added or changed, each application may need to be changed as well.
- Data is controlled by applications.
  This could be dangerous, for if an application fails to do the necessary checks, database integrity is in jeopardy.
- Possible performance penalty.

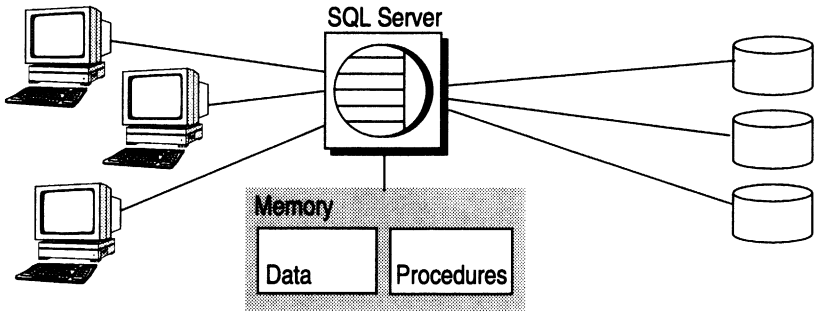
SYBASE SQL Server

You can program data integrity and enforce business rules on the SQL Server side. Advantages:

- Wherever you enforce business rules, database integrity can be guarded by programming SQL Server appropriately.
- If you program business rules into SQL Server, they are enforced consistently, no matter where data requests come from.
- Business rules are optimized for rapid execution and validation within SQL Server.
- Duplication of code in client application is eliminated for more efficient application development.
- You can choose to enforce rules on the application side to avoid network traffic or customize error messages.

Overview

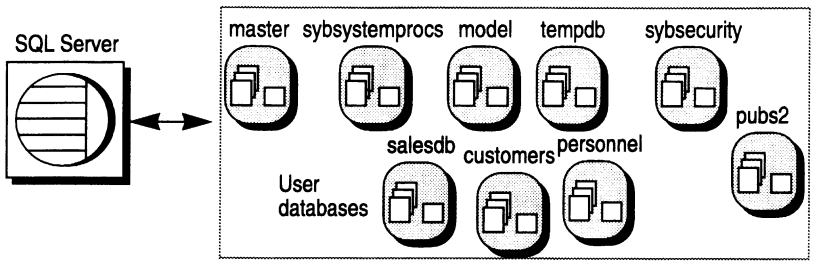
SQL Server Functions



- SQL Server manages user tasks, memory and data on physical devices
  - multiple users with multiple tasks each
  - multiple databases and their location on multiple disks
  - mapping of logical objects to data on physical devices
  - data and procedure caches in memory

SYBASE

Manages and Accesses Many Databases



- A database is a collection of tables holding related information.
- A single SQL Server can have multiple databases each of which can span multiple disks.
- Each database has its own options, object information, and transaction log.
- The SQL Server RDBMS knows the physical layout of the database so that it can find the data.

SQL Server Databases

master: Contains system tables

sybsystemprocs: Contains system stored procedures

model: Contains prototype database information

tempdb: Contains working storage and temporary tables

sybsecurity: Contains audit information (optional)

pubs2: Demonstration database (optional). **pubs2** is based on a fictitious book distribution company.

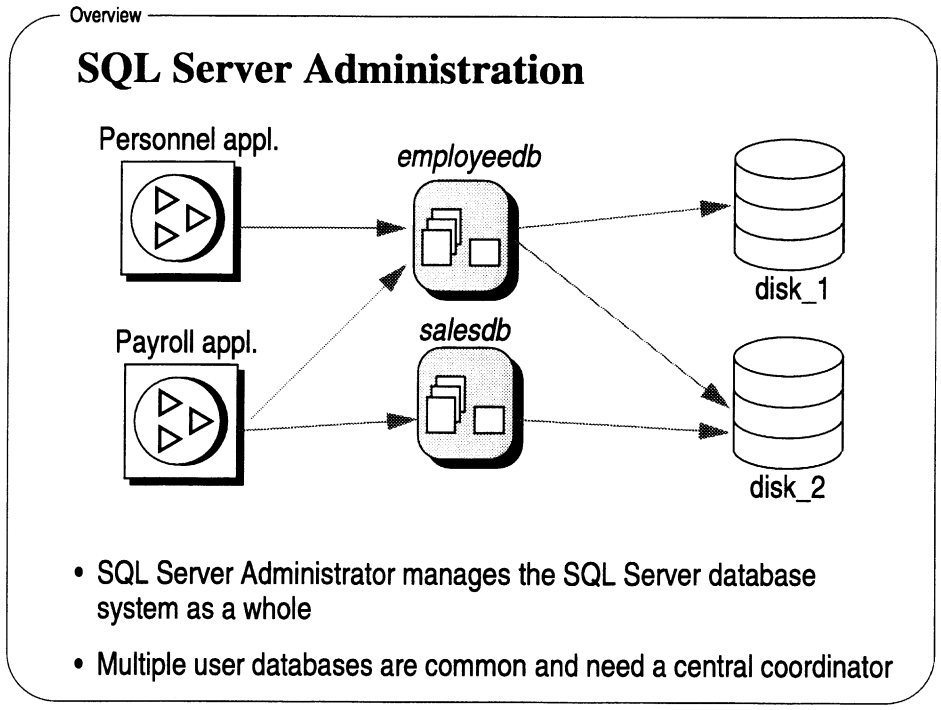
User Databases: User-defined application databases in production or development environment.

Overview

SQL Server Features

- High performance → **Fast**
- Server-enforced integrity and security → **Safe**
- Optimizer based on shortest processing time → **Smart**
- Multi-threaded → **Efficient**
- High availability → **Reliable**

SYBASE



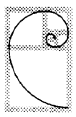
This foil shows sample applications accessing one or more databases on a SQL Server:

Personnel application Accesses *employeedb*, which is spread across two physical disks: *disk_1* and *disk_2*.

Payroll application Accesses both *employeedb* and *salesdb*. Notice that all of *salesdb* is on *disk_2*.

Administrative Responsibilities

Responsibility	Covered in Module:
Start, stop SQL Server	2
Allocate resources	3
Create, drop and enlarge databases	4
Create and manage SQL Server login accounts	5
Configure SQL Server	6
Maintain regular backups, restore as needed	8
Monitor, tune and troubleshoot SQL Server	9
Set up auditing for SQL Server	10

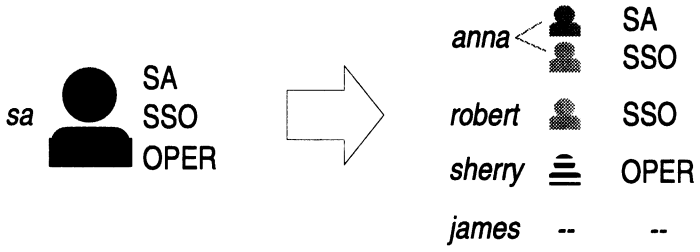
SYBASE 

The "*sa*" Login, & Three Special Roles

- The *sa* login is initially assigned three special roles:
  - **SA** System Administration (sa_role)
  - **SSO** System Security Officer (sso_role)
  - **OPER** Operator (oper_role)
- Each of these roles is authorized to perform certain administrative tasks

Overview

Dividing Up Administrative Responsibilities



- The *sa* login is initially assigned all three roles and is all-powerful
- To divide responsibility and increase accountability, you can assign these roles to separate logins and then "lock" the *sa* login
- These logins then become SAs, SSOs, or OPERs as appropriate, and can perform related functions

two choices

- have the *sa* login perform all system administration and security-related functions, as set up initially
- assign these roles to user logins and then lock the *sa* login

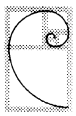
advantages of dividing up responsibilities

- can set up auditing (covered in a later module) and be able to determine, for example, *which* SSO user performed a certain security-related command

locking a login

- when a login is locked, one cannot log in
- a locked login still has a row in the *syslogins* table
- it can be "unlocked" (=reactivated) at any time by a login with the SA or SSO role

Locking and unlocking logins will be discussed in the module on Controlling Access.

SYBASE 

The Database Owner (*dbo*)

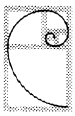
- The *dbo* "owns" the database and performs certain database-specific administrative functions to maintain it
  - Create tables and other objects
  - Add users and groups; define privileges
  - Back up and restore the database and transaction log
- It is a good idea for the *dbo* to own all the objects in a database, but this is not strictly necessary
- The *dbo* can become any user in the database by using the `setuser` command
- System Administrators automatically become *dbo* when they access a database

Overview

System Tables in Each Database

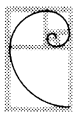
- Each database has system tables that contain information about objects and users for that database
- Some tables:

sysobjects	syslogs
sysprotects	sysprocedures
sysusers	syssegments
sysroles	sysalternates
systypes	syscolumns

SYBASE 

System Tables in Each Database (cont.)

- When you create a database object, modify an object, add a user, etc., the system tables are modified to reflect the new information
- Examples:
  - When you add a user, a row gets added to *sysusers*
  - When you grant permission to access a certain object, a row gets added to *sysprotects*
  - When you modify data in a table, rows get added to *syslogs*



SYBASE

Overview

System Tables in the *master* Database

- The *master* database contains:
  - system tables common to every database
  - additional system tables (see diagram) with server-wide information
- Some tables:

sysservers	syslogins
syssrvroles	sysdevices
sysdatabases	syslanguages
sysprocesses	syscurconfigs
sysmessages	sysconfigures

server-wide information

System tables in the *master* database contain information about all databases, devices, and SQL Server logins, about space allocation, configuration, remote Servers and logins, language for Server error messages, etc.

SYBASE 

System Tables in *master* (cont.)

- When you perform certain commands, the system tables are modified:
  - When you create a database, a row gets added to *sysdatabases*
  - When you change a password, a row in *syslogins* is modified
  - When you drop a device, a row is deleted from *sysdevices*
- Access to *master* should be limited for most users — "read-only" is sufficient
- Keep *master* free of user objects
  - to speed up recovery if you lose *master*
  - to avoid having *master* fill up

other system tables in *master*

There are many more system tables in master than have been listed on the previous foil. See the appendices to the System Administration Guide for complete details.

There are two system tables that do not begin with "sys": *spt_values* and *spt_committab*.

spt_values: contains, among other things, maximum and minimum values for Server configuration variables

spt_committab: contains information for Two-Phase Commit transactions. Details in the module on Other Server Administration Topics.

Both of these tables are not protected, i.e., can be updated. It is safer *not* to update them. In any case, do not delete them!

Overview

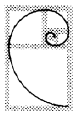
Stored Procedures

- Stored procedures are SQL statements that are stored in SQL Server
  - Some are supplied with the product — these are called system procedures
- System procedures typically access system tables
  - Display information:

`sp_helpdb`	*sysdatabases*
`sp_who`	*sysprocesses*

  - Modify tables:

`sp_addumpdevice`	*sysdevices*
`sp_addlogin`	*syslogins*

SYBASE 

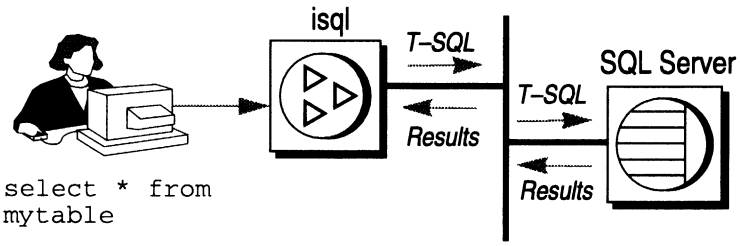
Updating System Tables

- Never update system tables directly!
  - Consequences of an error could be unrecoverable
- Use the system stored procedures to update system tables
  - Examples:
    sp_adduser, sp_addlogin, etc.

Overview

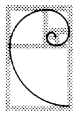
A Client: isql

- isql is an interactive SQL utility invoked at the OS command line level



  - Handles connection to SQL Server
  - Sends Transact-SQL to SQL Server
  - Displays returned rows and messages
  - Interfaces to the text editor on the OS

SYBASE SQL Server Utility Programs for your platform.

SYBASE 

Using isql

- Typical usage:

```
isql -U username <Return>
Password: <password>
> <some SQL statements>
> go
```

- The "go" command sends the batch of preceding statements to SQL Server
- Common commands:

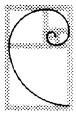
reset	clear commands and start again
vi or edit	call editor to edit last set of commands
quit	terminate a session, exiting from SQL Server

Script Files

- Script files are text files containing one or more batches of Transact-SQL code, each terminated by "go"
- Can be used to create or alter databases, add logins and users, create objects, set permissions, load small amounts of data, etc.

```
create table sales
(stor_id char(4) not null,
ord_num varchar(20) not null,
date datetime not null)
go

grant select on sales to public
go

create unique clustered index salesind
on sales (stor_id, ord_num)
go
```

- Example: *scripts/installpubs2* in sybase directory

SYBASE 

Using Scripts

- Use scripts as input files to isql:
  - Unix: `isql -Uabc -Pxyz -i filename`
  - VMS: `isql /u="abc" /p="xyz" /input=filespec`
- Create/update scripts in OS files as you create your databases, objects, users, permissions, etc.
- Treat these as source code--you may need them one day!

Summary

- System administration includes the following tasks:
  - Controlling and allocating disk resources
  - Creating and managing SQL Server logins
  - Managing system-wide security
  - Setting up backup procedures
  - Monitoring, tuning and troubleshooting
- The *sa* login owns the *master* database and (at installation) has all permissions
- SA, SSO and OPER roles can be assigned to specific logins
- System tables are critical to the functioning of SQL Server
- Save critical commands in script files; run them using `isql`

Lab 1a – Overview

By way of review, this lab asks you to list features of Client/Server architecture and list administrative responsibilities. Then we ask you to log into SQL Server, exit from SQL Server, write a script, and run it using isql.

The optional exercise asks you to familiarize yourselves with the system tables and write two queries against them.

1. List 3 features of Client/Server architecture:

   _____

   _____

   _____

2. List 5 administrative responsibilities:

   _____

   _____

   _____

   _____

   _____

3. Log into SQL Server using isql. (Ask instructor for instructions on username and password, and fill these in on the information sheet on the next page.) Run sp_helpdb to display the databases on your server.

4. Exit from isql. Write a short script which creates a table in *tempdb* called *testN*, where *N* is your user number. Run the script using isql. Log into SQL Server again, use *tempdb*, and display the table structure using *sp_help testN*. (Hint: Your script will have two batches: one with *use tempdb*, one which creates the table.)

Optional:

5. Consult the entity relationship diagram of system tables in the Appendix to this course and write select statements to:
   (a) display the text of error message 1205
   (b) using a join between two tables, list the names of the databases on your SQL Server alongside the login names of their owners

   Hints: For (a), check the contents of *sysmessages*. For (b), use *syslogins* and *sysdatabases,* joining on *suid.*

Lab Information Sheet

my servername: _____

my Operating System login: _____ (if required for your classroom)

my Operating System password: _____ (if required for your classroom)

my SQL Server login: _____

my SQL Server password: _____

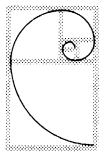
my default database: _____

How do I access the server?

```
isql -USQL_Server_login -PSQL_Server_password
```

Example:

```
isql -Ustudent42 -Pstudent42
```



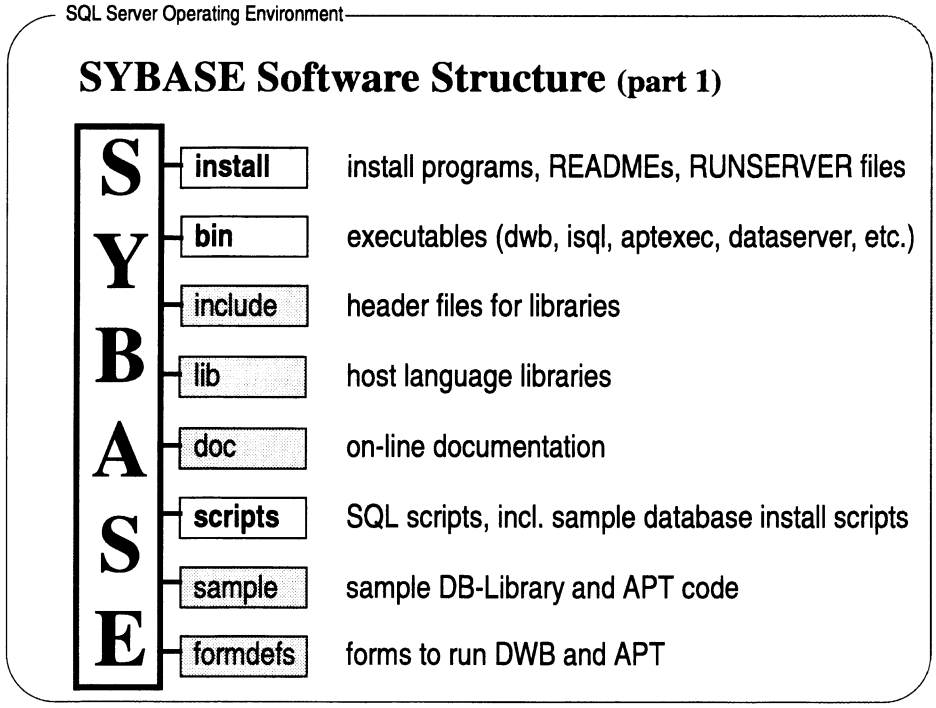
SYBASE®

Module 2

SQL Server Operating Environment

Objectives

- Describe SQL Server operating environment
- Describe what happens when you install SQL Server
- Identify parameters needed for installation
- Access SQL Server
- Start and stop SQL Server



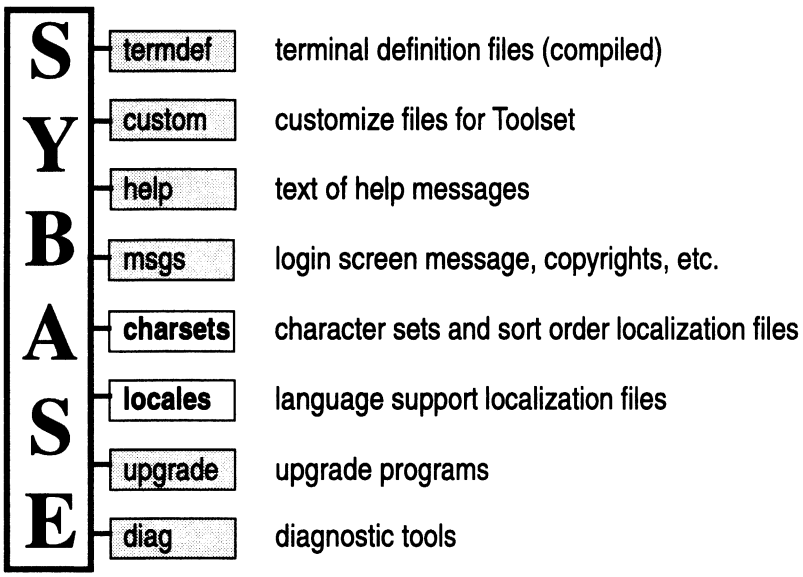
Global tapes

Tapes are global and contain the directories and files listed on this page and the next.

Your actual directory structure depends on just what you have purchased. This information is stored in your "Customer Authorization String".

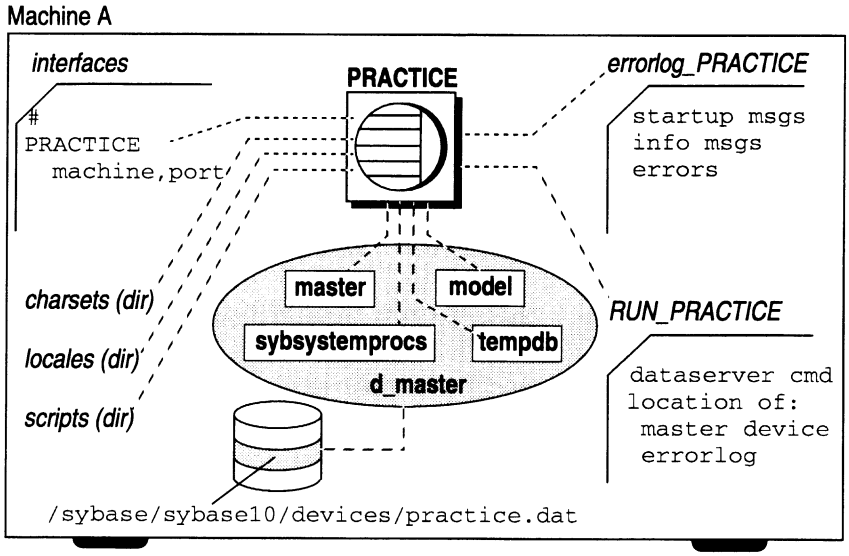
SQL Server Operating Environment

SYBASE Software Structure (part 2)



SYBASE

SQL Server Operating Environment



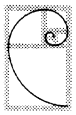
Installing SQL Server: What Happens?

`sybinit`, the installation program:

- Can install and start a Backup Server
- Creates or modifies *interfaces* file
- Creates *RUNSERVER* files
- Determines location of *errorlog* file (which gets created at SQL Server startup)
- Runs `buildmaster`, which
  - initializes the specified device as a master device
  - builds the master and model databases on that device
- Initializes sybsystemprocs device, if separate from master device

separate device for *sybsystemprocs*

To be discussed later in this module.

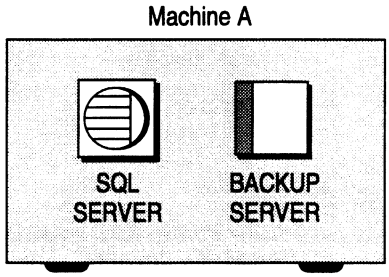
SYBASE 

Installing SQL Server: What Happens? (cont.)

- Starts SQL Server in the background, using the default character set and sort order for the platform
- Runs `installmaster` and `installmodel`
  - These are scripts which create/load system tables, system objects and stored procedures
- Installs
  - selected default sort order and character set (if different from defaults)
  - default language and additional languages
  - additional character sets

Installing a Backup Server

- The installation program can install a Backup Server on the same machine as SQL Server



- All SQL Server backups are done via this Backup Server

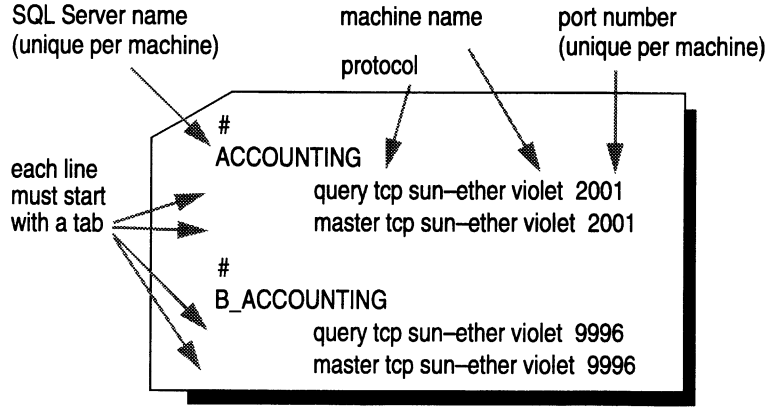
backups

We will talk in more detail about Backup Servers in the module on Backing Up Databases & Logs.

SYBASE

The *interfaces* File

- The installation program creates or appends to the *interfaces* file in the *sybase* directory
- Contains names, network addresses of SQL and Backup Servers

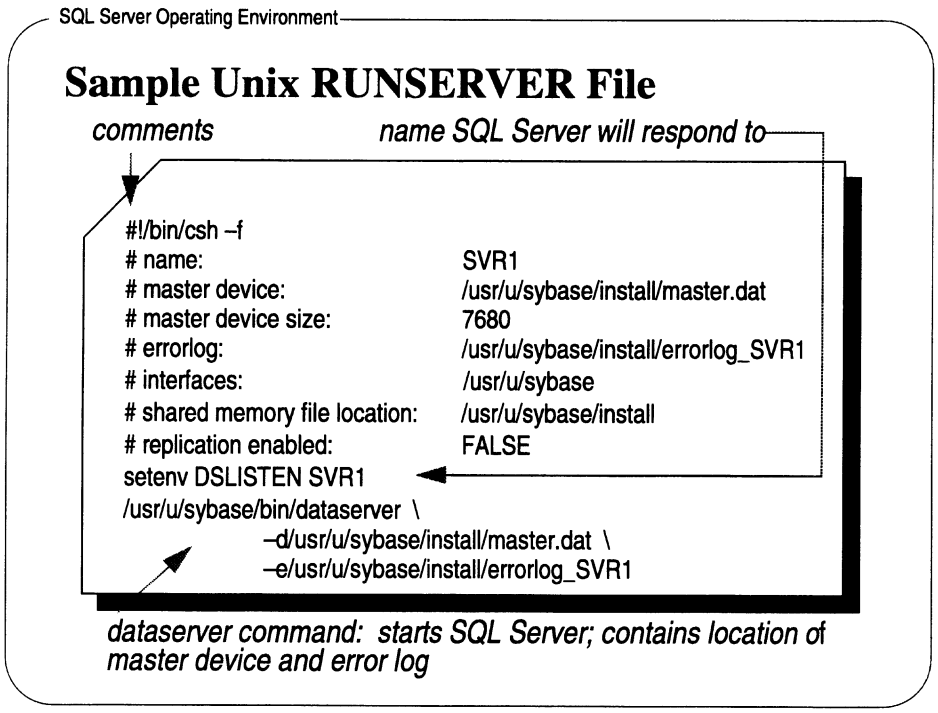


***interfaces* file**

- With the exception of the comment (#) and Server name line, all lines begin with a tab.
- Keep this in mind if you edit the file manually, and beware of e-mailing or cutting and pasting *interfaces* files!
- *query* line used by clients to find SQL Servers
- *master* line used by SQL Servers to determine port they should "listen" on (see upcoming discussion of DSLISTEN)

RUNSERVER Files

- RUNSERVER files are OS command files built by the installation program and used by `startserver` to start SQL Server or Backup Server
- One RUNSERVER file exists for each SQL Server and Backup Server installed on the machine
- Name (Unix)
  - RUNSERVER (If SQL Server is called "SYBASE")
  - Otherwise: *RUN_SERVERNAME*
  - Examples: RUN_SVR1, RUN_VIOLET, etc
- Name (OpenVMS)
  - RUNSERVER.COM (if SQL Server is called "SYBASE")
  - Otherwise: *RUN_SERVERNAME.COM*



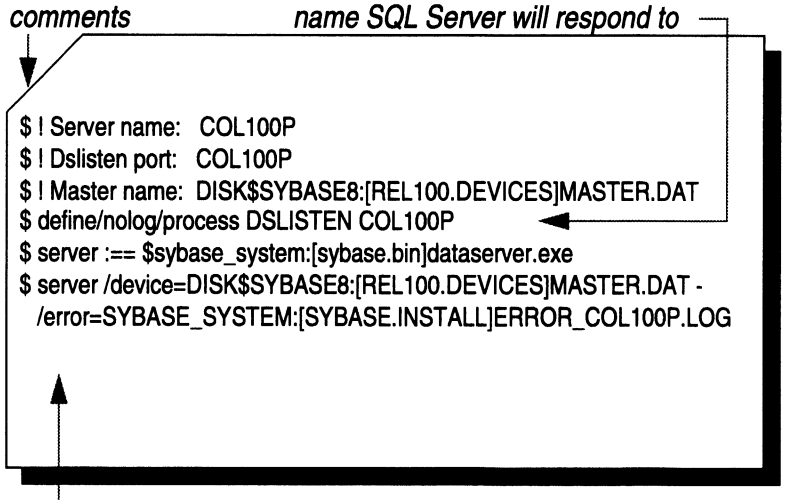
comments

- guaranteed accurate at installation
- subsequent changes not automatically reflected in RUNSERVER
- edit RUNSERVER file when you make changes to items in comments section in order to document your changes

dataserver

This is the command that gets generated by the install program. You may want to use other command line flags. If so, you can edit the RUNSERVER file.

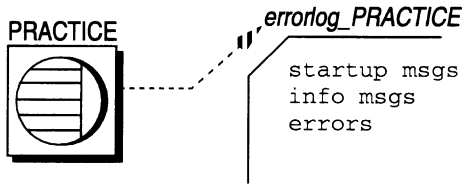
Sample OpenVMS RUNSERVER File



SYBASE

Creates or Appends to errorlog

- The errorlog file is a text file that logs informational and error messages from SQL Server

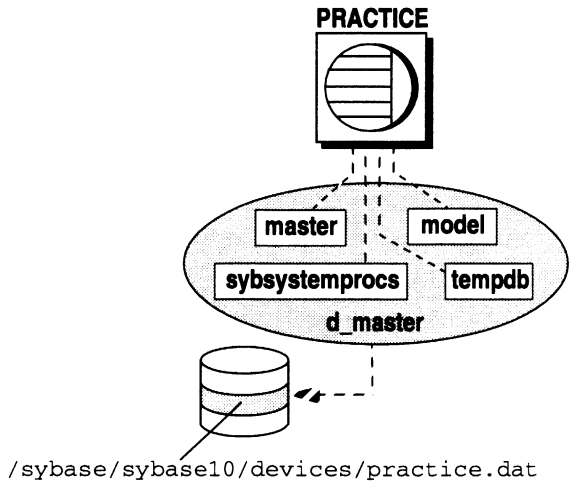


- During installation you can specify where the errorlog should go
  - Default: *install* directory
- It gets created the first time SQL Server starts (during installation) and is named errorlog_*SERVERNAME*

errorlog

More details in the module on Monitoring & Troubleshooting SQL Server.

Initializes Master Device



initializes master device

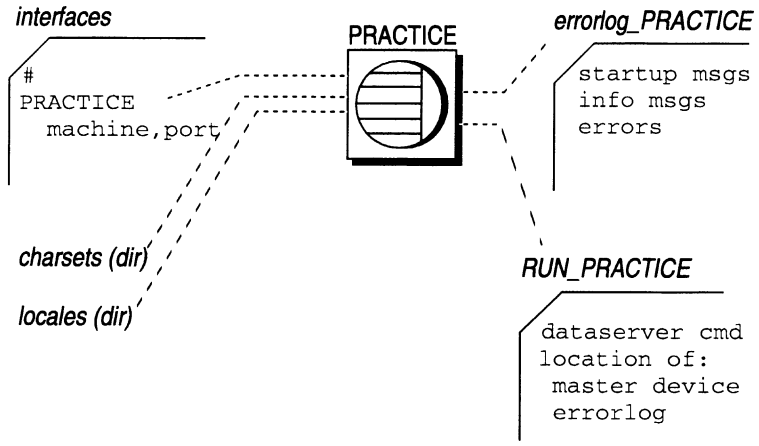
The installation program

- initializes the specified device as a master device
- installs the system databases (*master, model,* and *tempdb*) on that device
- installs *sybsystemprocs* on that device if no separate device has been specified

Note that `sybinit` can create a separate device for *sybsystemprocs* if you specify this during installation. This is recommended so there is room for expanding the *master* database on the master device.

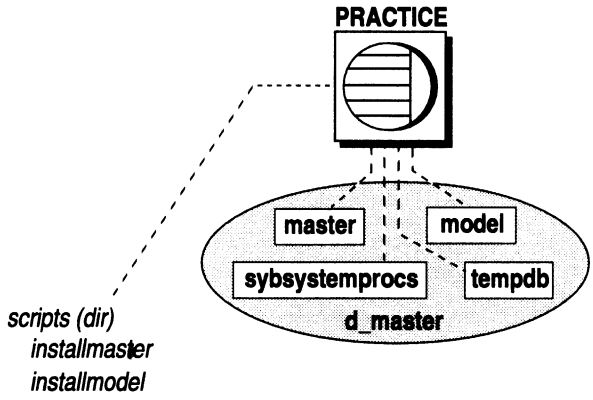
For simplicity, we show *sybsystemprocs* on the master device.

Starts SQL Server



- The installation program starts SQL Server in the background, using the default character set and sort order for the platform

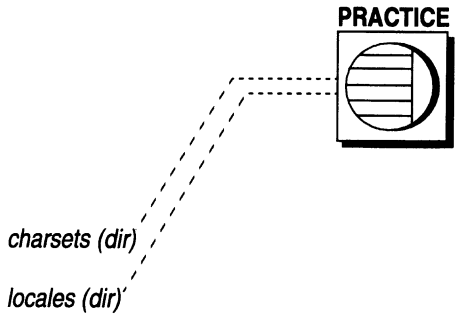
Runs *installmaster*, *installmodel* scripts



- These create and load system tables, system objects, and stored procedures into the *master*, *model*, *sybsystemprocs* databases

SYBASE

Sort Order, Character Set(s), Language(s)

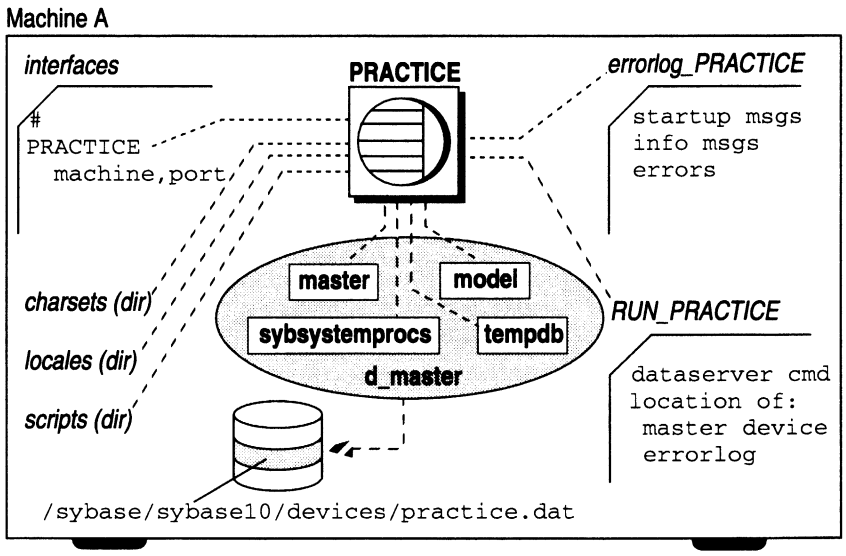


- The installation program installs select default sort order, character set, and language, and additional languages and character sets

System Administration Guide, Chapter 17: "Language, Sort Order, and Character Set Issues"

SQL Server Operating Environment

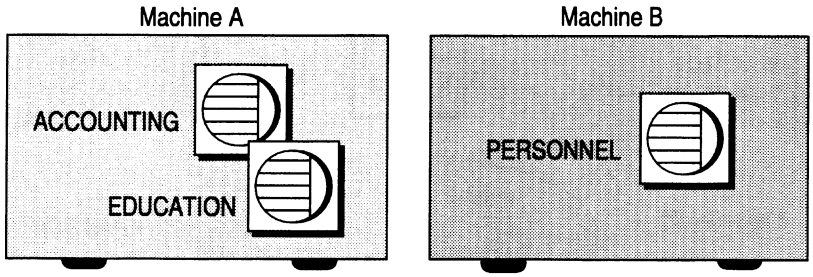
SQL Server Operating Environment



SYBASE

SQL Server Name

- SQL Server names should be unique across the network



- Default: SYBASE
  - We recommend naming a SQL Server after its function, department or major application

SQL Server name

- The Server name you choose at installation affects the contents and/or file names of *interfaces*, the *RUNSERVER* file, and the error log.
- The name assigned at installation is not stored internally in SQL Server, unless you also execute `sp_addserver` *`server_name`* to add the name of the current Server to the system tables. This is necessary only if you plan to implement remote procedure calls.
- To *change* the name of SQL Server after installation, simply change the name in all the relevant OS files.
- In most cases SQL Server names are unique throughout the enterprise, but for a specific application you may choose to use the same names on multiple machines.

Port Number/Object Name

- The installation program will prompt for port numbers for both SQL Server and the Backup Server
  - To find available port numbers, check OS configuration
- The port number will be reflected in the *interfaces* file and will help clients find servers and tell servers what port to listen on

```
#
ACCOUNTING
        query tcp sun-ether godzilla 2001
        master tcp sun-ether godzilla 2001
#
VIOLET_BKUP
        query tcp sun-ether godzilla 9996
        master tcp sun-ether godzilla 9996
```

Unix

To find available port numbers, check /etc/services file. After choosing a port number, be sure to modify this file to indicate what range of port numbers you are using.

OpenVMS

To find object names currently in use:
$ MCR NCP
NCP>SHOW KNOWN OBJECTS
NCP>EXIT

SYBASE

Size & Location of Master Device & sybsystemprocs Device

- Minimum size of master device: 17 Mb (see Installation Guide)
- What type of device should you choose?

Operating system	What kinds of devices to use
Unix	raw partition preferred, disk file O.K. for master device (if mirrored)
OpenVMS	RMS or foreign disk

- During development, you may choose a device that is less secure but easier to administer (e.g. disk files on Unix)

Unix

Unix disk files are not recommended for production user databases because Unix buffers writes, and there is no way for SQL Server to know for sure whether data was copied or not. Disk files may be used in development, however, as recoverability is less critical.

Disk files may be appropriate for the master device, even in production, as there is little update activity and it can be mirrored quite easily.

When choosing raw partitions, avoid cylinder 0, c, swap partitions, and any others which may be in use by the OS.

To find a raw partition, compare output for dkinfo and fstab.

OpenVMS

Both disk files and foreign disks will work well. In the case of foreign disks, the entire disk will be used.

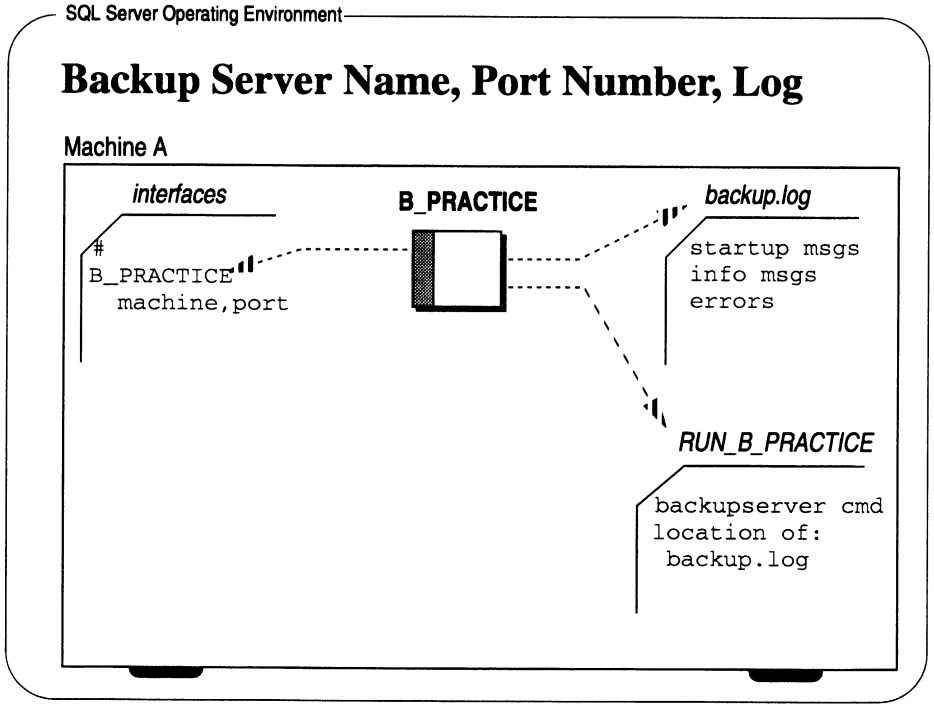
Disk files must be writable by *sybase* and must not already exist.

sybsystemprocs

During installation, the user will be asked to specify a device name for *master* and, optionally, *sybsystemprocs*. If no device name is specified for *sybsystemprocs*, this database is created on the master device.

It is a good idea to create a separate device for *sybsystemprocs* so there is room for expanding the *master* database on the master device.

Backup Server Name, Port Number, Log



Backup Server name

Backup Server names must be unique on every machine.

Default: SYB_BACKUP

To give Backup Server a name other than SYB_BACKUP, specify the name when prompted by the installation program.

port number

As with SQL Server, Backup Server requires a port number.

error log

Backup Server error logs must be unique on every machine. Default: *backup.log*. If there are more than one Backup Servers running on a machine, specify different names for each.

SYBASE

Unix Installation/Upgrade Worksheet (excerpts)

Record Server Information for Interfaces File

Server name (in 7-bit *ascii* characters): _____

Host Name: _____

Query port number: _____

Alias: _____

Choose Master Device Size

Size for master device in megabytes (>= 17Mb): _____

Choose Master Device Location

Raw Disk Partition Installation

Size of the raw partition in megabytes or pages:

_____

Name of partition: _____

❑ Partition deleted from */etc/fstab*.

❑ Owner and permissions changed.

OpenVMS Installation/Upgrade Worksheet (excerpts)

SQL Server Master Device Information

❑ **RMS File:**

❑ Directory for Master Device created

Directory Specification (including device):

_____

File name for Master Device File:

_____

Size for Master Device in Mb (at least 17 Mb):

_____

OR

❑ **FOREIGN Device:**

Name: _____

❑ FOREIGN Device Mounted

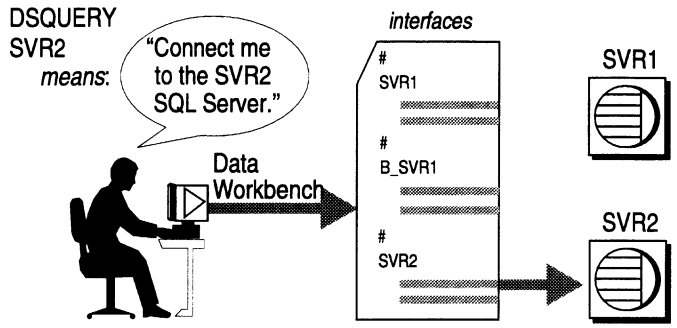
Name SQL Server

Name of SQL Server
[11 characters maximum; default is "SYBASE"]:

_____

SYBASE

Accessing SQL Server



- To connect to a SQL Server, specify its name using either DSQUERY or a flag on the command line
- The client program will look in the *interfaces* file for the named Server and read the *query* line to find its network address

isql:

-Usa
-Ppw
-Sservername ← !
(overschrijft DSQUERY parameter)

Shutting Down SQL Server

- Bring the SQL Server down gracefully with the SQL `shutdown` command
  - Minimizes the work that recovery must do when system is rebooted
- What `shutdown` does:
  - Disables logins except by System Administrators (SAs)
  - Performs a `checkpoint` in every database
  - Waits for currently executing SQL statements or stored procedures to finish
  - Prevents users (except SAs) from running any more SQL statements
  - Causes SQL Server process to exit

shutdown

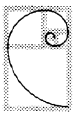
`shutdown` is the best way to bring SQL Server down.

shutdown with nowait

When this option is used, SQL Server shuts down immediately without waiting for currently executing statements to complete, and without performing a `checkpoint` in every database.

checkpoint

Checkpoints write log pages and changed data pages from cache (memory) to disk. We will be discussing checkpoints in the module on Transaction Management.

SYBASE 

Starting SQL Server

- To start SQL Server, go to the *install* directory and execute `startserver`
- Example:

```
cd  $SYBASE/install
startserver -f RUN_MYSERVER -m
```

- With no options, `startserver` uses the file "RUNSERVER" to execute appropriate `dataserver` command
- Options:
  - f to specify other RUNSERVER files
  - m to run single-user (SA) mode

startserver -f

`startserver` is located in `$SYBASE/bin` and you may wish to add this to your path.

Use when the RUNSERVER file is named something other than *RUNSERVER*, or to start the Backup Server running at the same time as the SQL Server:

```
startserver -f RUN_MYSERVER
       -f RUN_B_MYSERVER
```

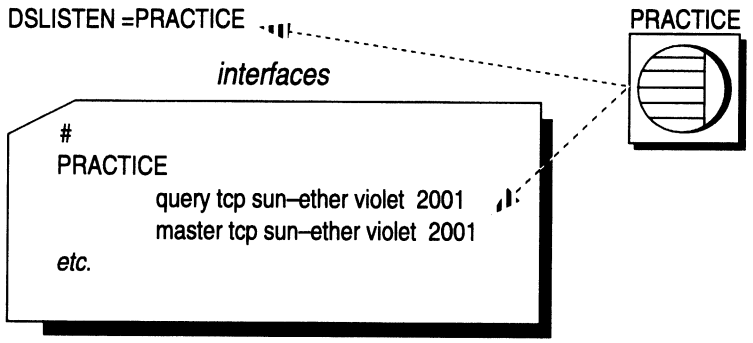
To start just the Backup Server:

```
startserver -f RUN_B_MYSERVER
```

startserver -m

"m" stands for "masterrecover". This option is used for restoring the *master* database.

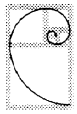
DSLISTEN



- When SQL Server is started, it checks the value of DSLISTEN to know its name
- It reads the *master* line of the *interfaces* file to determine what port to start on

setting DSLISTEN

DSLISTEN is set automatically in the RUNSERVER file.

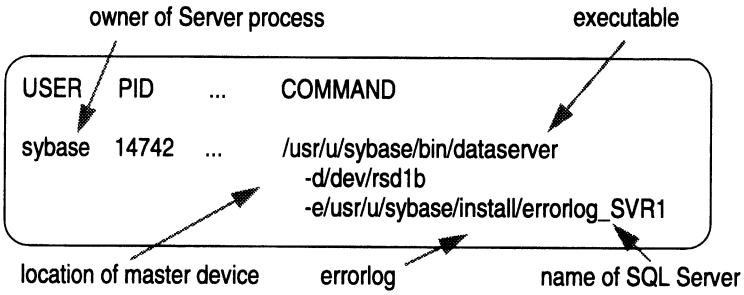


SYBASE

- The SYBASE environment variable/system logical should point to the *sybase* release directory
- Default location for all the SYBASE directories and files needed by both Clients and Servers:
  - *install, bin, scripts, charsets, locales*, etc.
- Sample uses:
  - Clients and Servers look there (by default) to find *interfaces* file
  - Servers look there to find language files, etc.

Displaying the Server Process

- The `showserver` command (in *install* directory) shows SQL Servers currently running on the local machine.
  - Sample output:



- You can also use operating system commands to display the SQL Server process

Unix

Execute `showserver` or `ps`.

OpenVMS

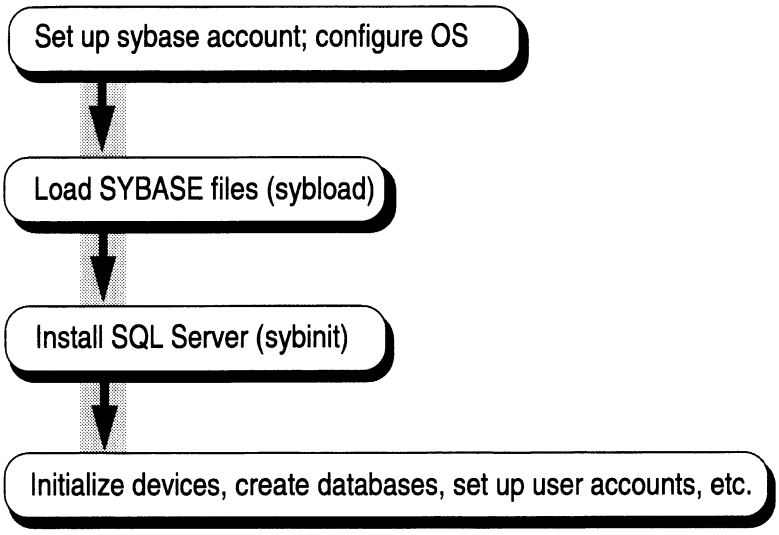
Execute `showserver` or SHOW SYSTEM.

Sample output:

Pid	Process Name	State	Pri	I/O	*etc.*
28E00081	SWAPPER	HIB	16	0	...
28E00086	CONFIGURE	HIB	10	76	...
...					
28E000A6	COL100P_SQL	HIB	6	28588	...

SYBASE

Installation Sequence



Things to Remember

- Choose location of the master device carefully
- Make sure the *sybase* account owns the master device
- Choose a port number that is not in use
- Start SQL Server as *sybase*, not as a superuser
- In the *interfaces* file, *query* and *master* lines must begin with a tab

Lab 2a - SQL Server Operating Environment

In this lab, we ask you to examine the sybase directory and certain files within it to determine what SQL Servers are installed and running. We ask you to draw a diagram with all the files related to one of these SQL Servers and answer certain questions about it. We then ask you to check the value of DSQUERY, access SQL Server, and display the Server name.

1. If you installed a SQL Server called VIOLET, what files would be created or modified?

2. Look in the *sybase* directory tree ($SYBASE in Unix and OpenVMS) to find which SQL Server or Servers are known to your system.
   a) Which SQL Server(s) are installed on your machine?
   b) Which are running?

3. For a SQL Server that is running, look through the files online and recreate a diagram similar to the one at the beginning of this module. Include the names of relevant files and their content (as it pertains to the SQL Server in question).

4. Answer the following using the RUNSERVER and interfaces files:
   a) Where is the master device, and how big is it?
   b) What ports is the Server using?
   c) Where is the errorlog?
   d) Which of these can you be certain is accurate?

5. Run `showserver`. What information does `showserver` give you?

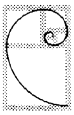
6. Scroll through the error log. Then return to your home directory.

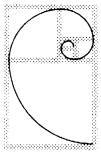
7. Check the value of DSQUERY (Unix: printenv DSQUERY; OpenVMS: SHOW DSQUERY)) and access that Server as *sa* using isql. The password is null.

8. Run `select @@servername`. Then exit isql.

9. What command would you execute to stop SQL Server? (Do not do this.)

10. What command would you use to start SQL Server again? (Do not do this.)





SYBASE®

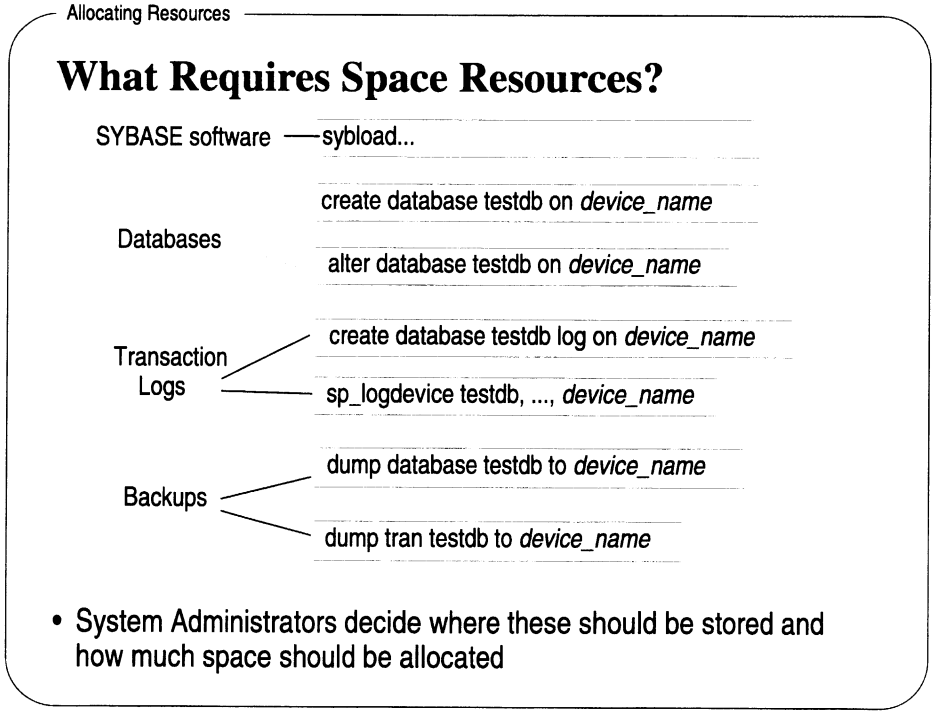
Module 3

Allocating Resources

Objectives

- Initialize devices for databases
- Describe the function and structure of *sysdevices*
- Display information about initialized devices
- Set up disk mirroring for database devices

SYBASE



backups

We will be talking in detail about backups in the module entitled Backing Up Databases & Logs.

What Kinds of Physical Devices to Choose

Operating system	What kinds of devices to use
Unix	raw partition preferred, disk file ***not*** recommended for user databases
OpenVMS	RMS or foreign disk

- During development, you may choose a device which is less secure but easier to administer (e.g., disk files on Unix)

Unix

Unix disk files are not recommended for production user databases because Unix buffers writes, and there is no way for SQL Server to know for sure whether data was copied or not. Disk files may be used in development, however, as recoverability is less critical.

Disk files may be appropriate for the master device, even in production, as there is little update activity and it can be mirrored quite easily.

When choosing raw partitions, avoid cylinder 0, c, swap partitions, and any others which may be in use by the OS.

To find a raw partition, compare output for dkinfo and fstab.

OpenVMS

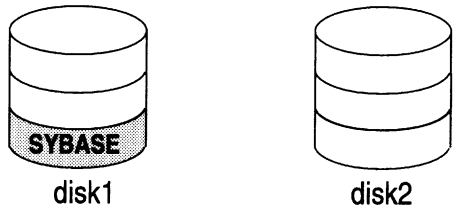
Both disk files and foreign disks will work well. In the case of foreign disks, the entire disk will be used.

Disk files must be writable by *sybase* and must not already exist.

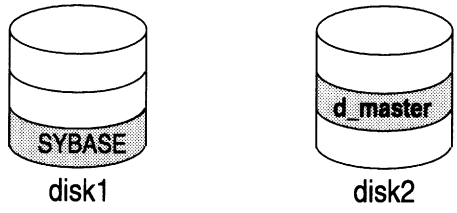
SYBASE

Placing SYBASE On Disks

- Unload SYBASE software...



- Install SQL Server...



unload SYBASE software

Unloading SYBASE software puts executables and other important files in the *sybase* directory somewhere on a physical disk. In this case, the SYBASE software is in a partition on *disk1*.

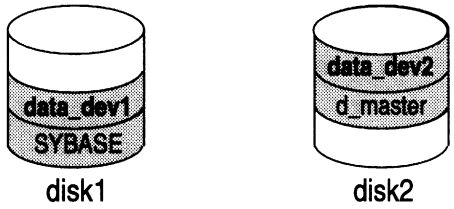
install SQL Server

When you install SQL Server, you affect certain files in the *sybase* directory tree. In addition, you create a *master* device containing *master*, *sybsystemprocs*, *model*, and *tempdb*. (You may choose to put *sybsystemprocs* on a separate device if your master device is not large enough.)

In this case, the *master* device is on a partition on *disk2*.

Placing SYBASE On Disks (cont.)

- Initialize devices...



initialize devices

In this step, you associate a logical name SQL Server will use to refer to the partition of a physical disk. Above, *data_dev1* and *data_dev2* are logical names SQL Server shall henceforth use to refer to certain partitions of *disk1* and *disk2,* respectively.

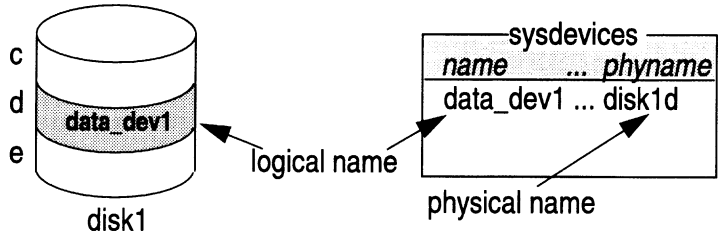
what next?

Create databases and segments on these logical devices. See the next module, "Creating Databases & Segments."

SYBASE

Initializing Database Devices

- A disk (or disk partition) must be made known to the SQL Server before it can be used for database storage
- `disk init` makes a physical disk or file usable by SQL Server and associates a logical name to the physical name



  - The logical name you assign will be used to refer to that device in subsequent SQL Server commands
- You can then create a database on that device

Unix

Raw partitions: SQL Server ensures that enough space is available and writes the allocation pages.

Disk files: the space required by that file is *not* preallocated--just the allocation pages are written. Make sure there is enough free space.

OpenVMS

All space required by the file gets allocated.

disk init: Basic Syntax

```
disk init
 name    = "device_name",
 physname = "physical_name",
 vdevno   = virtual_device_number,
 size = number_of_pages
 ...
 [,contiguous]  /* OpenVMS only */
```

name

Logical name of the device. Used in all references to this device from the SQL Server.

physname

Name of a raw partition or disk file. To create a disk file, sybase account must have write permission in the relevant directory. To use a raw partition, sybase account must own the relevant device. On Unix, the file cannot already exist.

vdevno

Virtual device number. Any number from 1 through (max configured - 1) that hasn't already been used. Check `sp_helpdevice` or *errorlog* for devices used. Check `sp_configure` (value of *devices*) for max configured. Limit: 255.

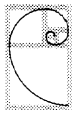
The master device is always vdevno 0. The highest vdevno that can be used is the value of *devices* - 1.

size

Size is in 2k pages. Tips: Specify sizes as multiples of 512 pages, that is, in whole megabytes.

contiguous **(OpenVMS only)**

Works with disk files only. Forces contiguous database file creation.

SYBASE 

disk init: Examples

- Examples:

```
Unix:
   disk init
   name = "data_dev1",
   physname = "/dev/rsd2d",
   vdevno = 2,
   size = 92160


OpenVMS:
   disk init
   name = data_dev1",
   physname =
   "disk$sybase8:[rel10.devices]sybex_dev.dat",
   vdevno = 3,
   size = 1024
```

sysdevices

- When you run `disk init`, a new row is added to the *sysdevices* table in *master*
- *sysdevices* lists database and dump devices
- Sample output of `select * from sysdevices:`

low	high	status	cntrltype	name	phyname	mirrorname
16777216	16782335	2	0	data_dev1	/dev/rsd2d	NULL
0	10239	3	0	master	d_master	NULL
0	20000	16	3	tapedump1	/dev/rmt4	NULL
0	20000	16	2	tapedump2	/dev/rst0	NULL

low, high

Virtual page numbers (except in the case of dump devices, where they represent capacity). Guaranteed to be unique for each database device. Calculated for you during disk initialization.

Device size (in pages) = high - low + 1.

status

The status column is a bit map indicating what kind of device it is, whether it's a default device, whether it is mirrored, and other useful information.

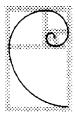
1 = default disk
2 = physical disk
4 = logical disk
8 = skip header
16 = dump device (tape or disk)
32 = serial writes
64 = device mirrored
128 = reads mirrored
256 = half-mirror only
512 = mirror enabled

The status bits are additive. For example, "3" indicates a physical disk that is also a default.

cntrltype

0 = database device
2 = disk or tape streaming dump devices
3-8 = tape dump devices

SYBASE



sp_helpdevice

- The `sp_helpdevice` procedure queries *sysdevices*, evaluates certain values, and prints out useful information about the devices
- Excerpt from output: `sp_helpdevice`

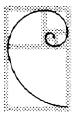
```
device_name    physical_name
   description
   status cntrltype device_number low     high
-----------------------------------------------
data_dev1      /dev/rsd2d
   special, physical disk, 10.00 MB
      2        0          1     16777216  16782335
master         d_master
   special, default disk, physical disk, 20 MB
      3        0          0        0         10239
tapedump1      /dev/rmt4
   tape, 625 MB, dump device
     16        3          0        0         20000
etc.
```

sp_helpdevice

- Without an argument, `sp_helpdevice` displays information about all devices on the Server.
- With an argument (a device name), `sp_helpdevice` displays information about that device only.

disk init: Notes

- Only System Administrators can execute `disk init`
- Before executing `disk init`:
  - Back up *master* database
  - Be sure there is enough disk space
  - Be sure the file or device hasn't already been initialized
  - Be sure *sybase* account has write permission on that device
  - Be sure SQL Server is configured properly with regard to devices, memory and connections
- Maximum configurable database devices: 255
- After executing `disk init`, back up the *master* database

SYBASE 

Default Devices

- When you create a database, you can specify which device it should be created on
  - If you do not specify, it is put on a *default* device
- After a device is initialized, System Administrators can set the device up as a default device using `sp_diskdefault`

```
sp_diskdefault logical_name,
  {defaulton|defaultoff}
```

- Example: Initialize *data_dev1*, then add it as a default disk:

```
disk init name = "data_dev1", physname =
     "/dev/rsd2d", etc.
exec sp_diskdefault data_dev1, defaulton
```

- `sp_diskdefault` marks these devices as default devices in *sysdevices*

default

Devices specified as default devices are used alphabetically when `default` or no device is specified in a `create database` statement.

Default Devices (cont.)

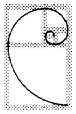
- When the SQL Server is installed, the *master* device is a default device
  - We strongly recommend changing this so that *master* doesn't get cluttered:

```
exec sp_diskdefault master, defaultoff
exec sp_diskdefault data_dev1, defaulton
```

- With separate `sp_diskdefault` commands, you can specify more than one default device
  - Multiple default devices will be used in alphabetical order as they fill up

why you should avoid putting other objects on *master*

In the event of a failure of the *master* device, recovery is complicated if there are additional objects on the device.

SYBASE 

Removing Devices

- When would you remove a device?
  - To change, repair, or add hardware
  - To change the size of a device (drop it; then add it back)
- To drop a device:

  `sp_dropdevice` *`device_name`*
  - Then restart SQL Server so you can reuse the device number (*vdevno*)
- In the case of disk files, you need to delete the file after dropping the device in order to reclaim the disk space
- You cannot drop a device if there are still databases on that device!

restart SQL Server after dropping a device

The kernel has a process that accesses a dropped device, and this cannot be killed without restarting SQL Server. Restarting frees up the logical device number for reuse.

Tips

- Use meaningful names
- Make sure the partitions, etc. you specify are really available, and be sure to follow platform-specific restrictions on what parts of a disk can be used
- Create and save scripts of your device allocation commands

initdatadevs

```
disk init
  ...
go
disk init
  ...
go
```

initlogdevs

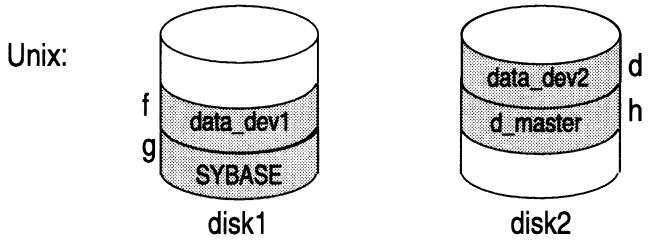
```
disk init
  ...
go
disk init
  ...
go
```

- Keep a hard copy of the *sysdevices* table

SYBASE

Tips (cont.)

- Keep a diagram of mappings of logical and physical devices



Lab 3a: Initializing Devices

In this lab, you will create two disk devices and make one of them a default device.

The optional exercise asks you to use the system tables to write a useful query. Given a logical device name, your query will return the corresponding physical device name.

See next page for syntax of useful commands for this lab.

1. Write and run a script which creates a disk device named *data_devN*, where *N* is your user number. The associated physical device should be a file called *data_devN.dat*. Make your device 4 MB. Use a *vdevno* assigned by the instructor.
   Reminder: Size is in pages, 1 page = 2K, 512 pages = 1Mb.
   Note: Your device will be in the *sybase/install* directory. At your site, you would probably put devices in a separate directory.

   script1.sql

2. Log into SQL Server using isql and use a stored procedure to check your results. Exit isql.

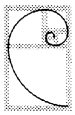
3. Write and run a script which makes the device which you just created a default disk. Check that your code had the desired effect.

4. Check if the master device is a default disk. If so, change it so that master is not a default device. Why is this a good idea? Check your results.

5. Write and run a script file which creates a second disk device called *log_devN*. Make it 2 MB, and map it to *log_devN.dat*.

Optional:

6. Write a query which given a logical device name will return the corresponding physical device name. Run `sp_helpdevice` to get some logical device names, and test your query.



Useful Commands for the Lab

Using isql to run script files:

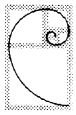
```
isql -Uloginname -Ppassword -i filename
```

Creating disk devices:

```
disk init
  name    = "device_name",
  physname = "physical_name",
  vdevno   = virtual_device_number,
  size = number_of_pages
```

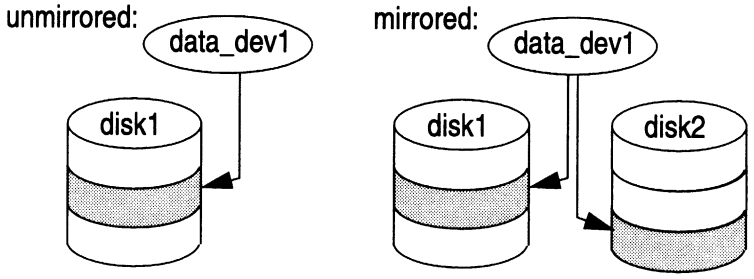
Making a device a default device:

```
sp_diskdefault logical_name, {defaulton|defaultoff}
```



Mirroring

- Mirroring is a feature which allows SQL Server to duplicate the contents of an entire database device
  - Writes go to both disks
  - Reads always come from the primary side



- If one disk for a mirrored pair fails, SQL Server notes this when it tries to read or write to that disk and continues with the other disk

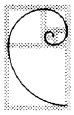
mirroring devices

- *entire devices* are mirrored, not databases
- to mirror a database, mirror all devices that database is on

System Administration Guide Chapter 3, Managing Physical Resources

SYBASE SQL Server Reference Manual Volume 1, Topics, Disk Mirroring.

SYBASE Troubleshooting Guide Chapter 5, Disk Mirroring.

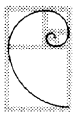
SYBASE 

Why Mirror?

- Benefits
  - prevent downtime due to disk failures
  - ensure full, non-stop recovery
- Costs
  - uses additional resources (disk storage)
  - writes are slower since they are duplicated
- Despite the costs, disk mirroring is highly recommended

What Should You Mirror?

- Mirror the most valuable and most vulnerable:
  - master device
  - log devices
  - active devices

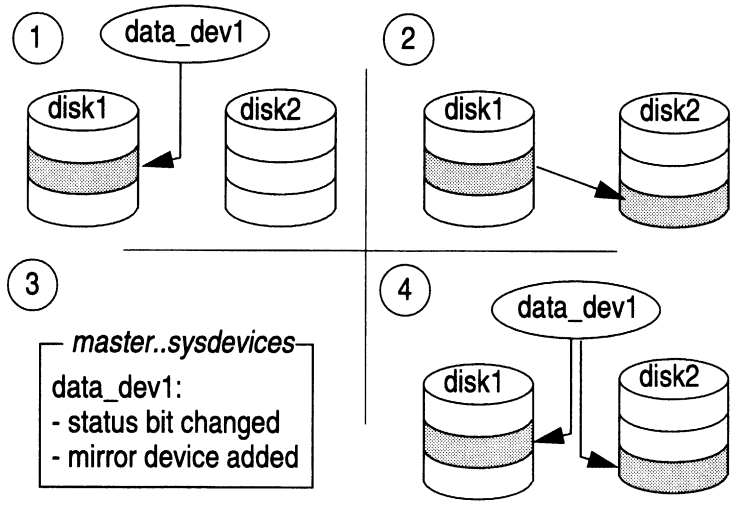
SYBASE 

How to Mirror a Device

- Use `disk mirror` to activate mirroring for a device
- Example:

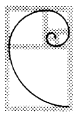
```
Unix:
   disk mirror
   name = "data_dev1",
   mirror =
      "/dev/rsd3d"

OpenVMS:
   disk mirror
   name = "data_dev1",
   mirror =
    "disk$sybase8:[rel10.devices]sybex_mir.dat"
```

What Happens When You Mirror a Device



What happens when you mirror a device

1. A secondary device is set up to mirror the primary device.
2. The entire contents of the primary device are copied to secondary device.
3. In the row describing that device in *sysdevices*, a bit in the *status* column is set and the physical name of the mirror device is recorded.
4. From this point on, the secondary device will be updated whenever the primary device is updated.

SYBASE 

disk mirror: Syntax

- Full syntax:

```
disk mirror
  name = "device_name",
  mirror = "physical_name"
  [ ,writes = {serial | noserial} ]
  [ ,contiguous ]  (OpenVMS only)
```

name	The logical name of the device you want to mirror
mirror	The physical name of the secondary (mirror) device
writes	Optional. Default is serial. On systems that support asynchronous i/o, serial guarantees write to primary device finishes before write to secondary begins. Serial is slower but safer: in case of failure, there is a greater chance that writes complete on at least one disk.
contiguous	Optional, OpenVMS only. Same as with disk init

Displaying Mirrored Devices

- The `sp_helpdevice` procedure displays mirroring information
- Sample output: `sp_helpdevice`

```
device_name   physical_name
  description
  status cntrltype device_number low    high
-------------------------------------------
data_dev1     /dev/data_dev1.dat
  special, physical disk, 10.00 MB
    2       0           1   16777216 16782335
log_dev1      /dev/log_dev1.dat
  special, MIRROR ENABLED,
  mirror='/dev/log_dev1_mir.dat', serial
  writes, reads mirrored, physical disk, 10.00
  MB
   738      0           2   33554432 33559551
master        d_master
  special, default disk, physical disk, 20 MB
    3       0           0       0        10239
etc.
```

device_name — logical name of the device

physical_name — physical location of the device

description — various pieces of information about the device: what kind of device is it, is it mirrored, if so, where is the mirror, etc.

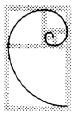
status — bit version of descriptive information

cntrltype — no longer used

device_number — virtual device number - *vdevno* as used by `disk init`

low — virtual page number of first page on device

high — virtual page number of last page on device
Note: high - low + 1 = size of device in pages

SYBASE 

Notes On Mirroring

- When you mirror a device, there is still just one logical device
- The secondary device must be at least as big as your primary device, and it should be on a separate physical disk
  - Be sure to follow platform-specific restrictions for mirroring unlike devices
- To mirror a database completely, mirror each device it is on
- Mirror disks when activity is low, and back up *master* when you are done
- Use `sp_helpdevice` to display information about mirroring

Unix: mirroring onto unlike devices

You can mirror files to raw devices, but it will be done using synchronous I/O. You cannot mirror raw devices to files since files do not support asynchronous I/O on most Unix systems.

SYBASE Troubleshooting Guide Chapter 5, Disk Mirroring.

Lab 3b: Mirroring

In the previous lab, you created two disk devices, one called data_devN and one called log_devN. In this lab, you will display details about log_devN, then mirror it, and then note how this affects the display.

Then there is a thought exercise: Which devices would you mirror given certain specifications?

The optional exercise asks you to consider your own business situation and decide which devices you will mirror.

1. Use a stored procedure to display details about *log_devN* (created in the last lab). Is the device mirrored?

2. Activate mirroring for that device. Call the file which will mirror your device "log_dev_mirrorN.dat".

3. Use a stored procedure to check your results. Which is the primary device, and which is the secondary?

4. Assume the following:
   Because of resource constraints, you can mirror only three of the devices listed below. Which would you choose to mirror?

A_dev	master device
B_dev	contains tempdb
C_dev	contains data from frequently-updated database1
D_dev	contains log from frequently-updated database1
E_dev	contains data from frequently-updated database2
F_dev	contains log from frequently-updated database2

5. For the above example, what if C_dev and D_dev were used for a "quiet" database? Would your answer change?

Optional:

6. In your own business situation, which devices will you choose to mirror?

Deactivating Mirroring

- Intentional or automatic
  - Intentional: by executing `disk unmirror`
  - Automatic: if an I/O error is detected on either disk
- What happens if there is an I/O error?
  - Status bits are set in *sysdevices* indicating deactivation, and which disk is still operational
  - I/O to the faulty device is disabled
  - `checkpoint` is performed on the *master* database
  - An error message is sent to the errorlog
  - Any `waitfor mirrorexit` processes are enabled

intentional deactivation

Use `disk unmirror`.

When might you do this? To reclaim the resource or perform hardware maintenance on other parts of the disk the mirror is on.

automatic

If an I/O error is detected, mirroring is automatically disabled. See foil for additional information.

checkpoint

Checkpoints will be discussed at length in the module on Transaction Management.

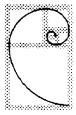
waitfor mirrorexit

In Transact-SQL, you can use `waitfor mirrorexit` to set up what processing is to occur whenever mirroring is disabled. This is usually done through a DB-Library program, but can be done using Transact-SQL.

Sample processing to notify SA or dbo of a mirror failure.

```
begin
   waitfor mirrorexit
   execute sendmail 'judy'
end
```

The `sendmail` procedure could insert a row into a certain table that the login *judy* could monitor for mirror failures.

SYBASE 

Summary of Mirroring Commands

- To mirror a device

```
disk mirror name = "device_name",
  mirror = "physical_name"
```

- To suspend mirroring on a device

```
disk unmirror name ="device_name"
```

- To resume mirroring on a device

```
disk remirror name ="device_name"
```

- To disable mirroring on a device permanently

```
disk unmirror name ="device_name", mode = remove
```

- To disable the primary device temporarily

```
disk unmirror name ="device_name",
  side = primary, mode = retain
```

- To reinstate primary device, use `disk remirror`

resume mirroring

When you resume mirroring, the first thing that happens is that the contents of the primary device are copied to the secondary device.

See System Administration Guide Chapter 3, and Reference Manual Volume 1, Topics. The Troubleshooting Guide also has a good discussion of what occurs in *sysdevices* as a result of each of the disk mirroring commands.

Supplying the Name of the Master Device Mirror at Startup

- If the master device is mirrored, edit *RUNSERVER* file to add the physical name of the mirror
  - If there is a problem with the primary master device, SQL Server will be able to access the mirror
- Unix:

```
dataserver -ddevicename ...
   [-rmastermirror_devicename]...
```

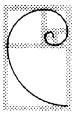
- OpenVMS:

```
dataserver [/device = (devicename
   [, mastermirror_devicename])]...
```

Why add the name of the master mirror?

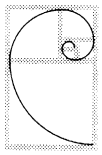
If primary is damaged and you try to start SQL Server, you can't access primary, therefore you can't access secondary.

Modify dataserver command in RUNSERVER file to instruct SQL Server where to look for secondary if primary is inaccessible. Edit comment section as well.

SYBASE 

Summary

- What requires device resources?
  - SYBASE software
  - databases (data and log), backups
- To add database devices, execute `disk init`
- To mirror devices, execute `disk mirror`
- To drop a device, execute `sp_dropdevice`
- To display information about devices, execute `sp_helpdevice` which reads from *sysdevices*



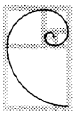
SYBASE®

Module 4

Creating Databases & Segments

Objectives

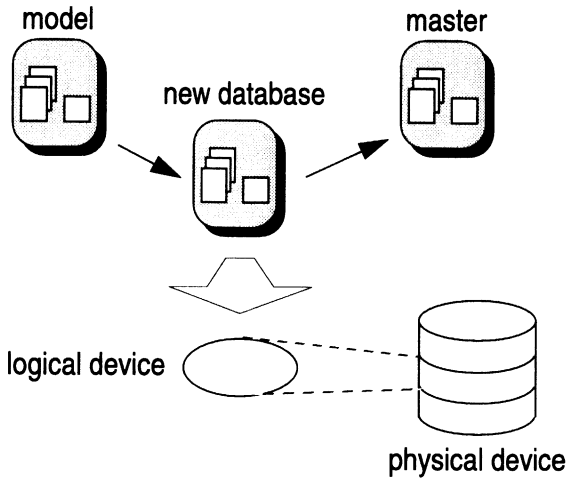
- Create databases
- Define segments and place objects on them
- Use three important system tables:
  - sysdatabases
  - sysdevices
  - sysusages
- Outline considerations for mapping databases to physical resources

SYBASE 

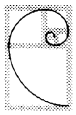
The Story So Far...

- Install SQL Server (initialize master device)
- Initialize database devices
- Set up mirroring
- Next steps:
  - create databases
  - add segments
  - create tables and other objects

What Happens when You Create a Database?



***model* database**	Used as a template to create the new database.
device(s)	Space reserved on device(s) for data and transaction log.
***master* database**	Entries for new database inserted into tables in *master* database.

SYBASE 

create database

- Basic syntax:

```
create database database_name
[on database_device [= size]
     [,database_device = size]]...]
[log on database_device = [size]]...]
```

- Examples:

```
(1) create database pubs2
(2) create database salesdb on data_dev1=5
(3) create database salesdb on data_dev1=5
        log on log_dev1 = 2
```

database_device

- If device not specified, database will be placed on a default device.

size

- If size not specified, size of model or configuration value will be used, whichever is larger.
- If there isn't enough disk space, SQL Server allocates what it can and sends message about how much it allocated. This must be at least the size of *model*.

Example 1

Data and log portions of the new pubs2 database will be on a single logical device, a default device (as none is specified). Size will be the size of *model* or of the configuration variable "database size", whichever is larger.

Example 2

Data and log portions of the new salesdb database will be on the data_dev1 device, size = 5 MB

Example 3

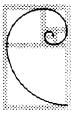
Data of the new salesdb database will be on data_dev1, size = 5 MB, and log will be on log_dev1, size = 2 MB. Total database size: 7 MB.

create database (cont.)

- Before creating databases, you need to decide:
  - what size to make the database
  - where to place it, and whether there is enough space available!
  - whether to create a separate log device, and if so, what size to make it
- You must be in the `master` database to execute `create database`

what you need to decide

We will be discussing database size and log issues over the next few pages, and general placement issues later in the module.

SYBASE 

Sizing a Database

- Size is in megabytes, minimum 2 MB
- Easy to expand, cannot be shrunk since space is preallocated
  - To make smaller, would have to recreate and copy data back in
- When estimating what size to make a database, consider mainly:
  - tables
  - indexes
  - transaction log
- Leave some free space, depending on anticipated activity
- Use `sp_estspace` to estimate the size of tables and their indexes.

sp_estspace

- **Basic syntax:**

  sp_estspace *table_name, no_of_rows*

- **Example, partial output:** sp_estspace titles, 10000

```
name       type        idx_level Pages  Kbytes
---------------------------------------------
titles     data              0     983    1966
titleidind clustered         0       7      14
titleidind clustered         1       1       2
titleind   nonclustered      0     279     558
titleind   nonclustered      1       9      18
titleind   nonclustered      2       1       2

Total_Mbytes:  2.50

(followed by summary data for titleidind,
titleind)
```

full syntax

```
sp_estspace table_name, no_of_rows
   [, fill_factor
   [, cols_to_max
   [, textbin_len
   [, iosec]]]]
```

table_name Name of the table. Must already exist, and must be in current database.

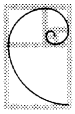
no_of_rows Estimated number of rows that table will contain.

fill_factor Index fillfactor. Default is null, meaning SQL Server uses its default fillfactor.

cols_to_max Comma-separated list of variable length columns for which to use the maximum length instead of the average.

textbin_len Length of all text and image columns, per row.

iosec Number of disk I/Os per second on this machine.

SYBASE 

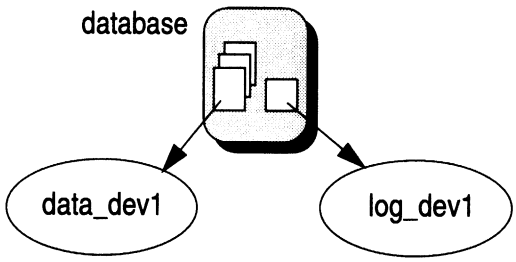
Sizing the Log

- Log size depends on activity (type and quantity of transactions) and on frequency of backups
  - Good starting point: 10-25% of overall database size
  - All inserts, deletes and updates are logged
  - For select into, and fast bulk copy: only space allocation and deallocation is logged
- Simulate running applications with the same number of users; then measure log usage using `dbcc checktable(syslogs)`
- The log is easy to expand, impossible to shrink

dbcc checktable(syslogs) See discussion in module on Monitoring & Troubleshooting SQL Server.

Placing the Log on a Separate Device

- Place the log on a separate logical device



  - Allows you to back up the transaction log separately
  - Provides protection against database filling up
  - Allows disk mirroring of the log for maximum uptime and up-to-the-minute recovery

- For performance & recovery, place log on a separate physical disk

placing the log on a separate physical disk

This will be discussed in greater detail later in this module and in the module on Backing Up Databases & Logs.

SYBASE

create database: Full Syntax

```
create database database_name
[on {default | database_device} [= size]
     [,database_device [= size]]...]
[log on database_device [= size,...]
     [, database_device [= size]]...]
[with override]
[for load]
```

- Examples:

```
(1) create database pubs2 on default = 4
(2) create database mydb on default = 3,
        data_dev1 = 2
(3) create database salesdb on sales_dev1 = 2
        log on sales_dev1 = 2 with override
```

- Note: We recommend putting the log on a separate device

default

Indicates that the new database can be put on any default device in *sysdevices* (as shown by *status*).

with override

Necessary if you use the "log on" clause to put the log on a separate device but use the same device name as for data, as in Example 3 above.

The "with override" clause forces SQL Server to accept these device specifications.

Up-to-the-minute recovery is endangered if you mix log and data on the same device.

for load

Invokes an abbreviated version of create database that can only be used for loading a database dump.

Use for recovery from media failure, or for moving a database from one machine to another. See module on Backing Up Databases & Logs.

Who Creates Databases?

- System Administrators (logins with the System Administrator role) can create databases
- The login that creates a database owns it initially

SYBASE

Transferring Database Ownership

- To transfer ownership of a database, access that database using `use` (in a separate batch!) and execute `sp_changedbowner`
- Syntax: `sp_changedbowner` *`login_name`*
  - The login in question must be a valid server login and *cannot* already be a user of the database
- Example:

```
use productsdb
go
sp_changedbowner fred
go
```

- Ownership of system databases cannot be transferred

system databases *master, sybsystemprocs, model, tempdb, sybsecurity*

Granting `create database` Permission

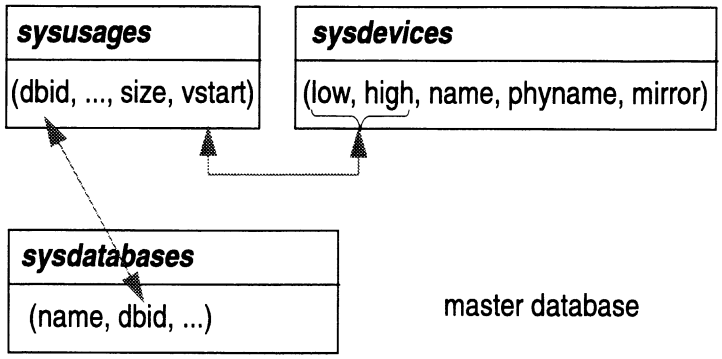
- System Administrators can grant `create database` permission to specific logins

```
grant create database to mary
```

- If such logins do not have the SA role, they must be users in the *master* database
- Control disk usage by restricting authorization to create databases

Displaying Database Location

- How does SQL Server know which device the database is on?



sysdatabases Lists database name, dbid, and other information.

sysusages Lists the device fragments for a given database.

vstart will fall between *low* and *high* of only one device *sysdevices*.

sysdevices Gives device name, physical name, mirror (if any), and and high page numbers.

Displaying Database Location (cont.)

- Sample selects from *sysusages*, *sysdevices*:

```
select * from sysusages where dbid =
db_id("smalldb")
```

dbid	segmap	lstart	size	vstart	pad	unresdpgs
5	3	0	1024	16777216	NULL	680
5	4	1024	512	33554432	NULL	496

- To display mapping for one of these device fragments:

```
select low, high, name, phyname, mirrorname
from sysdevices
where 16777216 between low and high
```

low	high	name	phyname	mirrorname
16777216	16782335	data_dev1	/syb/data_dev1.dat	NULL

segmap bit map of segment assignments. Discussed later in this module, in the section on Segments.

lstart first database (logical) page number.

size number of database (logical) pages.

vstart starting virtual page number

pad unused

unreservedpgs free space not part of an allocated extent.

SYBASE

Displaying Database Location and Usage

- `sp_helpdb` *`db_name`* displays location and disk usage for the specified database
- Sample output: `sp_helpdb smalldb`

name	db size	owner	dbid	created	status
smalldb	3.0 MB	sa	5	May 5, 1993	no options set

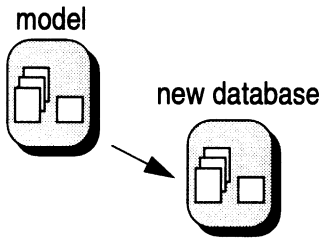
device fragments	size	usage	free kbytes
data_dev1	2.0 MB	data only	1376
log_dev1	1.0 MB	log only	1008

sample output

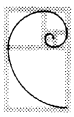
This output shows that the *smalldb* database is 3 MB large and is split across two devices: a 1 MB log and a 2 MB data fragment.

Customizing Databases

- When you create a database, the contents of *model* get copied to the new database



- You can customize model to contain the stored procedures, tables, rules, user-defined datatypes, users, privileges, options, etc. that you want all future databases to have
- Only System Administrators should be allowed to update *model*

SYBASE 

Setting Database Options

- Using `sp_dboption`, the following database options can be set:

```
abort tran on log full
allow nulls by default
dbo use only
ddl in tran
no chkpt on recovery
no free space acctg
read only
select into/bulkcopy
single user
trunc. log on chkpt.
```

- SSOs can enable/disable free space accounting; all the others are set by the database owner (or SAs acting as dbo)
- Database options for *master* cannot be changed

abort tran on log full — Abort transactions when last-chance threshold crossed.

allow nulls by default — When true, null values are allowed in a column even if not specified in column definition.

dbo use only — When this is set to "true", only dbo can use the database.

ddl in tran — Data definition statements allowed in transactions.

no chkpt on recovery — No checkpoint record added to log after recovery.

no free space acctg — Suppresses free-space accounting and execution of threshold actions for non-log segments.

read only — Users can retrieve database data but can't modify data.

select into/bulkcopy — Enables writetext, select into, and "fast" bulk copy.

single user — When true, only one user at a time can access database.

trunc. log on chkpt. — Transaction log truncated whenever the checkpoint checking process occurs.

Setting Database Options (cont.)

- Syntax:

```
sp_dboption  [dbname, option_name,
{true|false}]
```

- Usage example:

```
use master          /* must be in master */
go
sp_dboption "smalldb", "read only", true
go
use smalldb              /* must use database and... */
go
checkpoint               /* checkpoint the database */
go
```

- Any user can use `sp_helpdb` to display options currently set

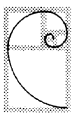
quotes

Quotes are needed around options to `sp_dboption` only if the option contains embedded blanks. It doesn't hurt to use them, however.

checkpoint

Refreshes in-memory structure called *dbtable*, which is what SQL Server uses to determine which option is currently enabled. Also forces all "dirty" pages (pages that have been updated since they were last written) to be written to the database device.

Discussed in module on Transaction Management.

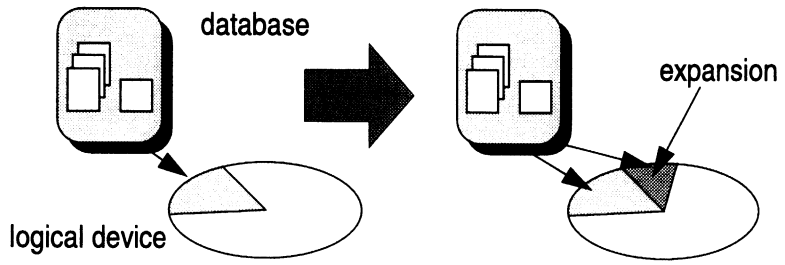
SYBASE 

Monitoring Space Usage

- Databases (either data or log segments) can fill up as users use a database
  - If this happens, all data modifications are suspended
- You can use these tools to monitor space usage in the database:
  - sp_helpdb
  - sp_helpsegment
  - sp_spaceused
  - threshold manager
- We will look at these in detail in the module on Monitoring & Troubleshooting SQL Server

What To Do If You Run Out of Space

- If you run out of space in the log, you have to truncate it
  - Possible fix: dump/truncate more frequently
- If you run out of space in the data segments, try to reclaim space by dropping unused objects
- Another alternative: expand the database (data and/or log)



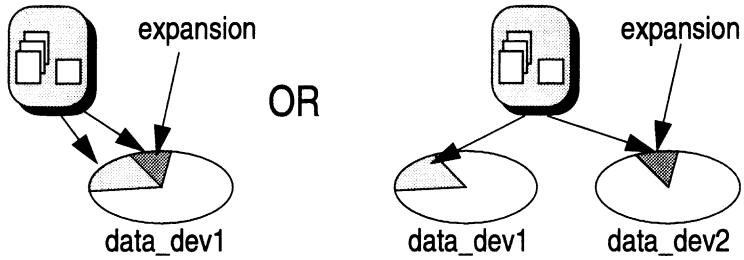
dumping and truncating the log See module on Backing Up Databases & Logs; also Monitoring & Troubleshooting SQL Server.

reclaiming space See Monitoring & Troubleshooting SQL Server.

SYBASE

Expanding a Database

- Using `alter database`, Database owners and System Administrators allocate additional space to a database on the same or different devices



- Database size cannot be decreased

expanding a database

Can be expanded onto the same device or onto a new device.

database size cannot be decreased

Space is pre-allocated and therefore cannot be taken back. The advantage: SQL Server can be maximally efficient since it optimizes allocation for its own use.

To "shrink" a database, you have to copy out all the data, drop the database, recreate it, and then load the data back in.

Expanding a Database (cont.)

- Basic syntax:

```
alter  database database_name
   [on {default | database_device} [= size]]
   [log on database_device [= size]]
```

- Size specified is increment of growth; default 1 MB
- Examples:

```
alter database pubs2
alter database pubs2 on data_dev1 = 3
alter database pubs2 on default = 2
```

- Sample sequence: create - alter - alter

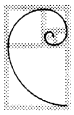
```
create database salesdb on data_dev1 = 5
alter database salesdb on data_dev2 = 2
alter database salesdb on data_dev3 = 1
```

full syntax

```
alter  database database_name
   [on {default | database_device} [= size]
   [, database_device [= size]]...]
   [log on { default | database_device }
         [= size]
   [, database_device [= size]]...]
   [with override]
   [for load]
```

size

The size you specify is the desired *increase*, not the new size of the database.

SYBASE 

Expanding a Database (cont.)

- You can expand a database while it is in use
- You can expand the *master* database only on the master device
- If you expand *tempdb* or *model* and you rebuild the master device, you will have to expand them again
  - Do not make *model* larger than *tempdb*

Separating the Log By Moving It to a New Device

- For a database created without using the "log on" option (that is, without a separate log), take the following steps:
  - dump the log to truncate it
  - alter the database onto the new device
  - execute `sp_logdevice` to make that device a log device
- Perform these steps with the minimum amount of time elapsed between each to avoid log records being written to the old device
- Effect:
  - no new pages will be allocated on the old device
  - what is there will be deallocated as the log gets truncated
- Result: the log will grow onto the new device

Note on sp_logdevice

sp_logdevice is *only* intended to be used to *separate* the log from the data, if the log and data are originally on the same device.

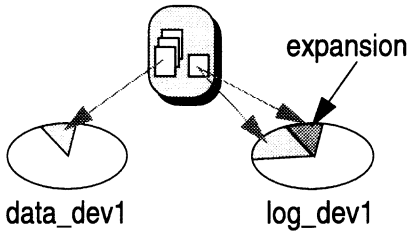
Refer to the System Administration Guide, Chapter 3 "Managing Physical Resources" for further information.

SYBASE

Expanding the Log

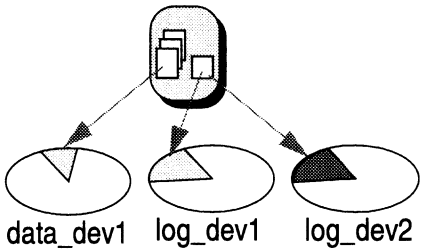
- On the same device:

```
alter database pubs2
  log on log_dev1 = 1
```



- Onto a new device:

```
alter database pubs2
  log on log_dev2 = 4
```



On the same device

The original database is created as:

```
create database pubs2 on data_dev1 = 10
   log on log_dev1 = 2
```

Data will be put on *data_dev1* and *syslogs* will be on *log_dev1*.

Here we expand the log for the database on the same device, *log_dev1*.

Onto a new device

The original database is created as:

```
create database pubs2 on data_dev1 = 10
   log on log_dev1 = 2
```

Data will be put on *data_dev1* and *syslogs* will be on *log_dev1*.

Here we expand the log for the database onto a new device, *log_dev2*.

sp_logdevice

Now we turn *log_dev2* into a log device, and *syslogs* moves there. *log_dev1*, formerly a log device, is now used for data only.

If you did not use the `log on` extension when you created the database, you can user `alter database` with the `log on` clause, then `sp_logdevice` to have separate data and log devices.

Dropping Databases

- Database owners and System Administrators can remove a database by executing `drop database`
- Syntax:

  ```
  drop database database_name
  ```

- Database must *not* be in use
- When to drop a database:
  - To remove experimental or old databases, reclaiming space
  - Prior to recovering a damaged database

Lab 4a: Creating Databases

In a previous lab, you initialized two devices: one called data_devN (4 MB) and another called log_devN (2MB). You set up mirroring on log_devN.

In this lab, we ask you to create a 3 MB database with the log on a separate device. We ask you to display information about this database, display all its objects, change an option, then reset it. See next page for syntax of useful commands for this lab.

The optional lab asks you to write two queries against sysdatabases, sysdevices, and sysusages--tables that contain critical information about databases and the devices they occupy. We also ask you to recreate create database statements given device fragment information.

1. Write and run a script in which you create a 3 MB database called *dbN*, where N is your user number. Place the data (2 MB) on *data_devN* and the log (1 MB) on *log_devN*.

2. Using a stored procedure, display information about this database, including size, placement, ownership. Now find your database in *sysdatabases*.

3. Display all the objects in your database. Besides system tables, what objects does the database contain? Can you influence what objects a newly-created database has? If so, how? (Don't do this, however!)

4. Make the database for dbo use only. Using a stored procedure, verify that your change took effect. Then undo this (set "dbo use only" back to false).

Optional:

5. Write a query to show which databases are on your device *data_devN*. Now run your query again for the master device. (Hint: Comparing *sysusages.vstart* with *sysdevices.low* and *sysdevices.high* will tell you which device each database fragment is on.)

6. Write a query to determine how much free space there is on your device *data_devN*. (Hint: The relevant rows in sysusages can be found using the same comparison as in #5 above.)

7. Given the following output from `sp_helpdb sales`, write the commands which would recreate *sales* if it were lost:

```
name  db_size owner dbid  created     status
sales 4 MB    sa    5     Oct 16 1992 no options set

device_fragments  size   usage          free kbytes
dev1              2 MB   data only            1376
dev2              1 MB   log only             1008
dev3              1 MB   data only            1008
```

Useful Commands for the Lab

Creating databases:

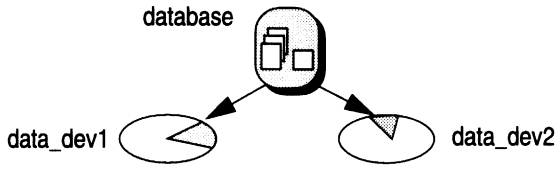
```
create database database_name
[on database_device [= size]
     [,database_device = size]]...]
[log on database_device = [size]]...]
```

Setting database options:

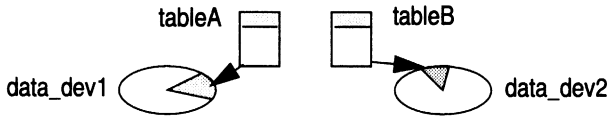
```
sp_dboption  [dbname, option_name, {true|false}]
```

Placing Objects on Devices

- A database can span several devices



- You may want to ensure that a certain large, high-use table is on one device and another large, high-use table is on another device

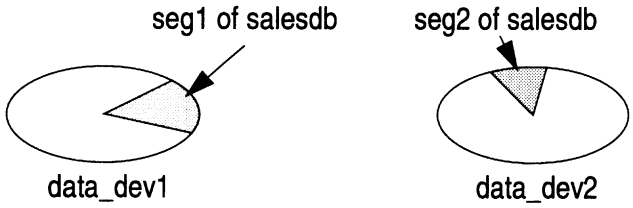


- To do this, use user-defined segments

SYBASE

Segments

- Segments label space on one or more logical devices for one database



- If you create a user-defined segment, you can place tables or indexes on that segment

```
create table tableA(...) on seg1
create table tableB(...) on seg2
```

- By controlling their location, you can arrange for active tables and indexes to be spread across disks

Segments

If a database has many fragments on a given logical device, then all fragments will be associated with the same segments.

Why Use Segments?

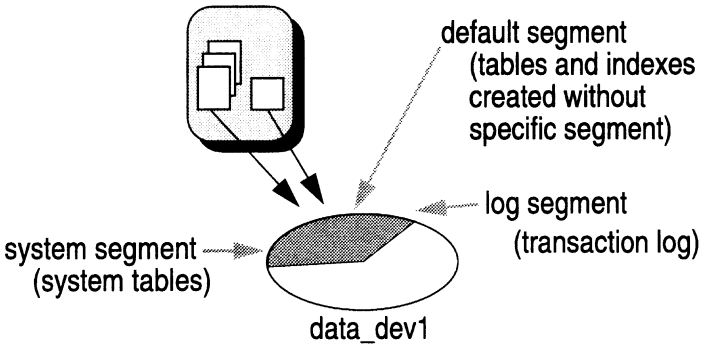
- Can improve performance
  - Splitting large tables across disks improves i/o throughput
  - Separating tables and their nonclustered indexes across disks improves i/o throughput
- Can control space usage
  - A table can never grow larger than its segment allocation; you can use segments to limit table size
  - Can take advantage of threshold manager to monitor space usage

threshold manager See module on Monitoring & Troubleshooting SQL Server.

SYBASE

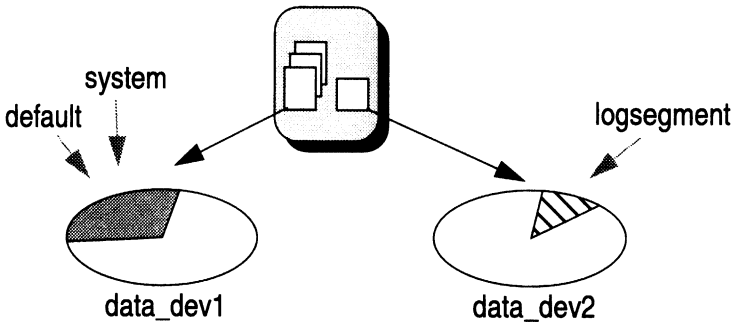
System-Defined Segments

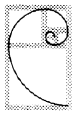
- When you create a database, space allocated is labelled with three system-defined segments: *system, default, logsegment*
- Sample database: data and log on same device



System-Defined Segments (cont.)

- Sample database: data and log on separate devices



SYBASE 

Adding a User-Defined Segment

- To define a segment, execute `sp_addsegment`
- Syntax:

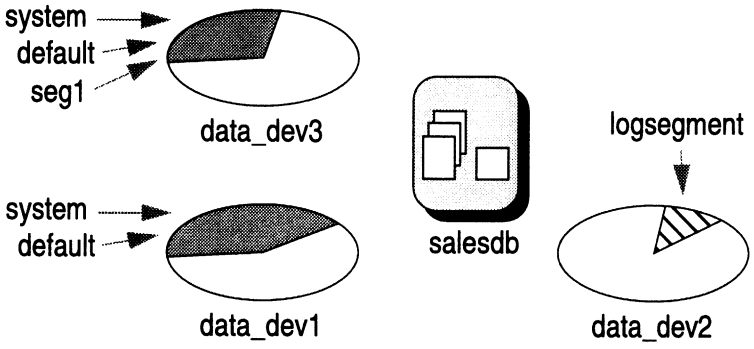
```
sp_addsegment segname, dbname, device_name
```

- Sample sequence:

```
disk init name = "data_dev3",...
go
alter database salesdb on data_dev3=1
go
use salesdb
go
sp_addsegment seg1, salesdb, data_dev3
go
```

- Defining a segment doesn't allocate additional space--it labels space that has already been allocated (above, by `alter database`)

Adding a User-Defined Segment (cont.)

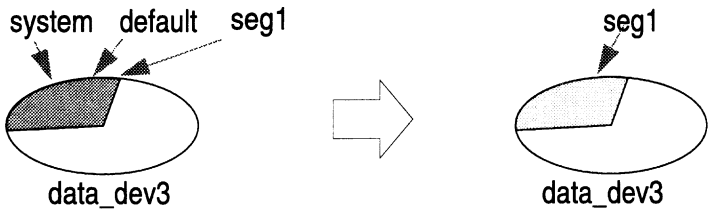


- Notice: after the `alter database` and `sp_addsegment` commands, *data_dev3* has *system*, *default*, and *seg1* segments of *salesdb*

SYBASE

Dropping System & Default Segments

- To ensure that objects you place on a user-defined segment do not have to compete with system tables and other objects, execute `sp_dropsegment` to drop the system and default segments from that device



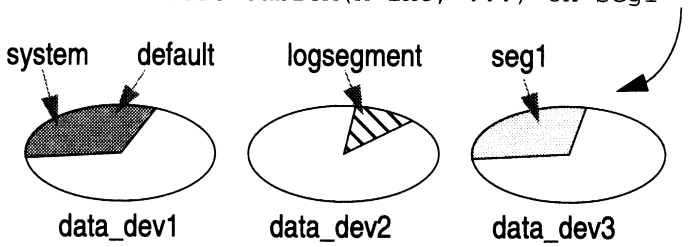
- Example:

```
sp_dropsegment system, salesdb, data_dev3
sp_dropsegment "default", salesdb, data_dev3
```

Creating Objects On Segments

- To place an object on a segment, specify the segment when creating the object

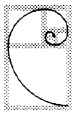
```
create table tableA(x int, ...) on seg1
```



- If you do not specify a segment name, the table will be placed somewhere on the default segment (which may span devices)
- Clustered indexes and their tables are always on the same segment

clustered indexes and tables

If you create a clustered index on a segment that is *not* the segment the table is on, the table moves to that new segment.

SYBASE 

Displaying Information About Segments

- Execute `sp_helpsegment` to list all the segments for the current database
- Sample output:

```
segment   name        status
    0     system      0
    1     default     1
    2     logsegment  0
    3     seg1        0
```

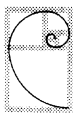
- Segment names are specific to the database
  - They will not be confused with similarly-named segments in other databases
- Segments are listed in order of their creation

status The default segment has a status of 1.

Displaying Information About Segments and Devices

- Execute `sp_helpdb` *`databasename`* when using the database to display the segments for that database
- Sample output:

```
name db_ size owner dbid created     status
sales 4 MB    sa    5    Oct 16 1992 no options set

device_fragments  size   usage          free kbytes
dev1              2 MB   data only            1376
dev2              1 MB   log only             1008
dev3              1 MB   data only            1008

device                  segment
dev1                    default
dev1                    system
dev2                    logsegment
dev3                    seg_1
```

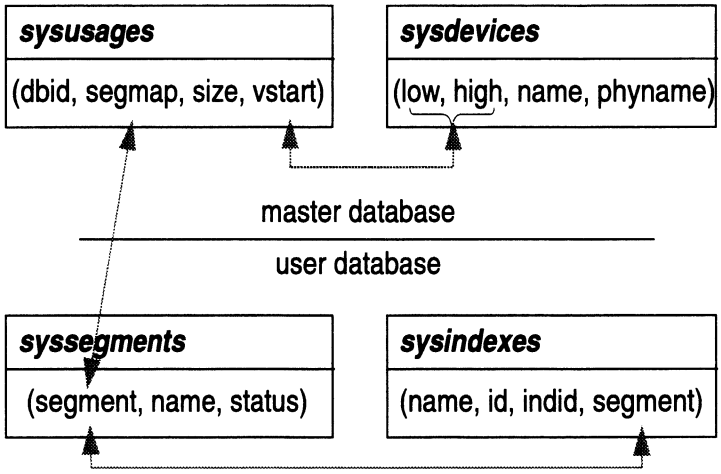
SYBASE 

Displaying Information About Objects on Segments

- Execute `sp_helpsegment [segname]` to list the objects on the segment and show what device(s) the segment is mapped to
- Sample output:

```
sp_helpsegment seg1

segment name    status
   4 seg1            0

device                size                free-space
data_dev3             1.0MB                      688

table_name          index_name      indid
tableA              x_tableA            1
```

System Tables Related to Segments



sysusages Contains a row for each device fragment assigned to a database. This is mapped to physical disks via *sysdevices*: *sysusages.vstart* falls between *sysdevices.low* and *sysdevices.high.*

See next page for a discussion of *segmap*.

syssegments Contains a row for each segment (named collection of disk pieces). Maps segment name to segment number.

sysindexes Contains a row for each index and for each table that does not have a clustered index. *segment* column indicates segment that will be used for future allocation for the object.

SYBASE

Segmap

sysusages (in *master*)

dbid	*segmap*	*lstart*	*size*	*vstart*
...	...	...	...	...
5	3	0	1024	50331648
5	4	1024	512	67108864
5	8	1536	512	83886080

... 8 4 2 1
0 ... 0 0 0 0 0 1 1
31...6 5 4 3 2 1 0

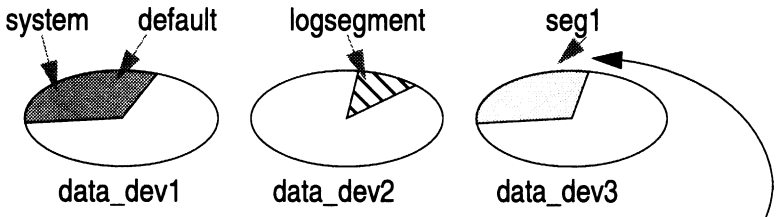
syssegments (in local database)

segment	*name*	*status*
0	system	0
1	default	1
2	logsegment	0
3	seg1	0

- *sysusages.segmap* is a decimal representation of a bitmap
- Using this value, you can determine which segments are mapped to the fragment the row describes

Our Database So Far...

- We have three devices for the *salesdb* database, with segments defined as follows:

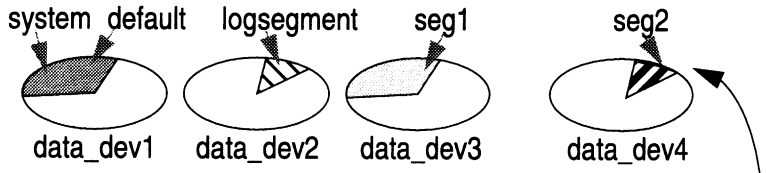


  - Large, heavily-used table *tableA* has been placed on *seg1*
- What can we do if we have another large, heavily-used table?
- What can we do if *tableA* outgrows *seg1*?

SYBASE

What If We Have Another Large Table?

- Solution: create a new segment on a new device

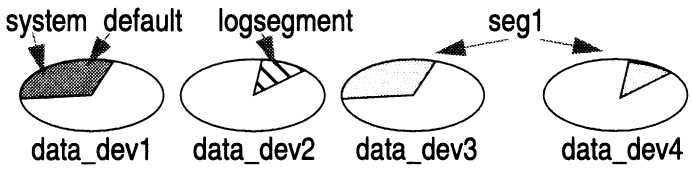


- Commands:

```
disk init; alter database; sp_addsegment;
sp_dropsegment (system and default);
create table...on seg2
```

What If There Is Not Enough Disk Space?

- What can we do if *tableA* outgrows the space *seg*1 can use?
- Solution: extend *seg1* to another device



- Commands:

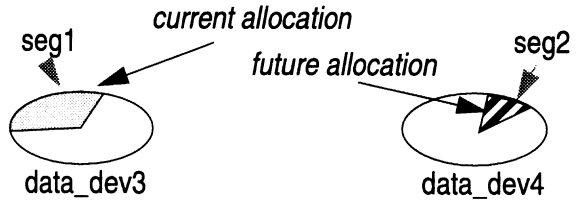
```
disk init; alter database; sp_extendsegment
  (onto dev4); sp_dropsegment (system and
  default)
```

SYBASE

Changing Segments for Existing Objects

- You can direct where tables or indexes will grow by executing
  `sp_placeobject segname, object`
- Example

```
sp_placeobject seg2, tableA
```



  - When *tableA* needs more space, SQL Server will use *seg2*
- Can be useful to split a large table across two segments or devices

what sp_placeobject does

- Changes the segment associated with the object so that all *future* expansion will occur on the new segment.
- Existing allocations remain where they are while the data remains unchanged.

steps in splitting an existing table evenly across two segments

To split tableA equally across two segments:

- Create two *new* segments (e.g., new_1 and new_2).
- Create a copy of tableA on one of the segments (e.g., create table newtableA(...) on new_1.
- Bulk copy half the data.
- Use sp_placeobject on the table so that new data goes to the other segment (new_2).
- Bulk copy the remaining data (which will automatically go to new_2).
- Drop the original table (tableA) and rename the new table (sp_rename newtableA tableA).

You now have restored the table in its entirety, and it is spread evenly across new_1 and new_2. To take care of future activity:

- Add a third new segment (e.g., new_3) on the same device as new_1, extend it to the device new_2 is on, and use sp_placeobject to ensure future writes go to new_3. SQL Server will allocate the space as needed.

Tips

- Give your segments names that identify what they are used for, and use extensions like "_seg"
- Save these commands in scripts!
- Diagram things out
- If possible, keep things simple by assigning one segment to one device
- Strike a balance between performance benefits of segments and the added complication in planning and administration

Lab 4b: Segments

In the previous lab, you created a 3 MB database called dbN, with data on data_devN and log on log_devN. In this lab, we ask you to arrange for your database to span two devices so that you can put one table on one device and the non-clustered index for that table on the other device. See upcoming page for syntax of useful commands for this lab.

We also provide excerpts from hypothetical sysdevices, sysusages, and syssegments tables. We ask you to use these to determine certain features of the database reflected therein.

1. Put the following commands in logical sequence first, and then execute them:
   _6_ Create a non-clustered index for *mytable* on *seg1*
   _3_ Add a segment called *seg1*, and place it on *index_devN*
   _2_ Alter your database so part of it is on *index_devN*
   _5_ Create a table called *mytable* on the default segment
   _1_ Add a 1 MB device called *index_devN*
   _4_ Drop *system* and *default* from *index_devN*

   Sample create table statement:
   ```
   create table mytable (a int, b int)
   ```
   Use a unique *vdevno* (provided by instructor) for the disk init statement, and put your device on a file named *index_devN.dat*. Use `sp_helpsegment` to verify your results.

2. Assume the following: There has been a system failure and the devices on which dbid 4 resided are unrecoverable. For some reason you have no backups and no scripts. Fortunately, you have up-to-date hard copies of the system tables.
   Looking at the excerpts of *sysdevices*, *sysusages*, and *syssegments* on the next page, answer the following questions:
   (a) How many device fragments are in the database whose *dbid* is 4?
   (b) How big are these fragments?
   (c) Which devices are they on?
   (d) What segments are mapped to each device?

Optional:

3. The rows in *sysusages* are in chronological order, according to how databases, etc. were created. Given that, determine what sequence of statements would be required to recreate the lost database.

(continue to next page)

Lab 4b: Segments (cont.)

master..sysdevices (excerpts):

low	high	status	cntrltype	name	phyname	mirrorname
67108864	67109887	2	0	contacts_dev	/usr/DEV/contacts_dev.dat	NULL
50331648	50334207	2	0	data_dev	/usr/DEV/data_dev.dat	NULL
33554432	33555455	2	0	log_dev	/usr/DEV/log_dev.dat	NULL

master..sysusages:

dbid	segmap	lstart	size	vstart	pad	unreservedpgs
1	7	0	1536	4	NULL	0
1	7	1536	1024	3588	NULL	0
1	7	2560	1024	4612	NULL	568
2	7	0	1024	2564	NULL	680
2	7	1024	512	8708	NULL	512
3	7	0	1024	1540	NULL	680
4	3	0	2560	50331648	NULL	1328
4	4	2560	1024	33554432	NULL	1024
4	8	3584	1024	67108864	NULL	1024
5	3	0	2560	5636	NULL	1392
5	4	2560	512	8196	NULL	400
5	7	3072	512	9220	NULL	0

syssegments (in the database):

segment	name	status
0	system	0
1	default	1
2	logsegment	0
3	contacts_seg	0

Useful Commands for the Lab

Creating indexes:

```
create index indexname on tablename(column_name) on segname
```

Adding segments:

```
sp_addsegment segname, dbname, device_name
```

Altering databases:

```
alter database database_name on device_name = size
```

Creating tables:

```
create table tablename (column_name datatype
  [, next_column datatype ]) [ on segname ]
```

Creating devices:

```
disk init
name   = "device_name",
physname = "physical_name",
vdevno  = virtual_device_number,
size = number_of_pages
```

Dropping segments:

```
sp_dropsegment segname, dbname [, device_name]
```

Summary of Segments

- Create user-defined segments to partition the space allocated to a database

```
use database database_name
sp_addsegment  segname, dbname, device_name
sp_extendsegment segname, dbname, device_name
```

- Place objects on segments to improve performance and control object size

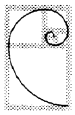
```
create table tablename(..) on segname
create index indexname on tablename on
     segname
sp_placeobject segname, object
```

- Display information; drop segments

```
sp_helpsegment [segname]
sp_helpdb dbname
sp_dropsegment segname, dbname
   [, device_name]
```

things to remember

- A clustered index and its table are always on the same segment
- The log is placed on a segment using the `log on` option to `create database`
- Segments are local to a database and can only include devices already allocated to the database

SYBASE 

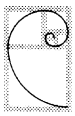
Disk Allocation Considerations

- The best disk allocation for a particular site will depend on:
  - how much data there is
  - what kinds of queries will be run against SQL Server
  - what resources you have available
  - what performance you require
  - how critical continuous uptime is
  - how you plan to address recovery
- Goal: to balance the load of disk accesses across available disks
- Disk allocation will affect:
  - performance
  - uptime
  - recoverability

Decisions You Need to Make

- ✓ What kinds of physical devices to use
- How much disk space you will need for:
  - ✓ data
  - ✓ indexes
  - ✓ logs

  tempdb
  backups
  sybsecurity/sysaudits (optional)
- Where to place each of these
- Whether to mirror devices
- Whether to use segments

SYBASE 

Sizing *tempdb*

- SQL Server uses *tempdb* as working storage space for queries involving:
  - `order by` (when different from index order)
  - group by
  - distinct
  - explicitly created #temp tables
- If SQL Server does not have the space it needs, the query fails
- To size *tempdb*, estimate use of these queries and number of users likely to be running them simultaneously
- Alter *tempdb* onto devices other than *master*

tempdb

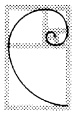
tempdb (like all databases) cannot be shrunk. If you need to decrease the size of a database, you must drop it and recreate it. In the specific case of master, sybsystemprocs, model, and tempdb, you need to rebuild the master device. See SYBASE Troubleshooting Guide for details.

Sizing Backup Devices and *sybsecurity*

- Backup devices
  - You can back up entire databases (data + log) or just the logs (if logs are on separate devices)
  - Allocate as much space for backups as for the data and/or log device
- sybsecurity/sysaudits
  - If you intend to audit activity, you will need to install the *sybsecurity* database and the *sysaudits* table
  - Size of *sysaudits* depends on how much activity you expect to be tracking

Backups — See module on Backing Up Databases & Logs.

***sybsecurity* database; *sysaudits* table** — See module on Auditing.

SYBASE 

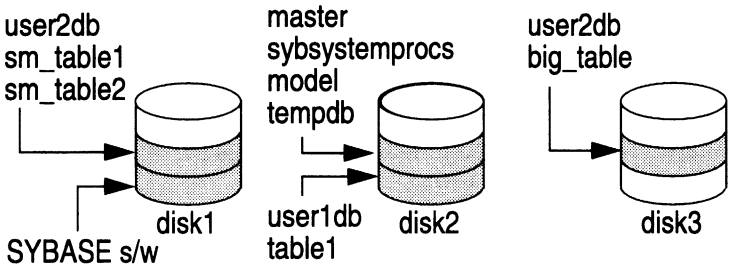
Where to Place Data

- On a small system or in a development environment, you might choose to put your entire database on the same physical disk, especially if it is easy to recreate from scratch
- Advantages
  - Simple
  - Uses minimal resources
- Disadvantages
  - Not fully recoverable in the case of a media failure (only good to the last full backup)
  - Performance suffers if tables are large and activity is heavy

Where to Place Data (cont.)

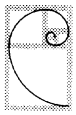
With larger tables, and where recoverability and performance are critical:

- Spread databases out evenly across disks, to minimize contention
  - place databases on separate disks
  - place large tables on separate disks using segments



master device

For simplicity, we show *sybsystemprocs* together with *master*, *model*, and *tempdb* on the master device. It is a good idea to put *sybsystemprocs* on a separate device from master so there is room left for expanding the *master* database on the master device.

SYBASE 

Where to Place the Log

- On small system or in a development environment, you may choose to store the log with the data on the same physical disk
- Advantages:
  - Simple
  - Uses minimal resources
- Disadvantages
  - Cannot do transaction log dumps
  - In case of a media failure, cannot be recovered up to the minute; only good to the last full backup
  - With high activity, performance suffers, as log writes have to compete with data writes

There are three possible scenarios for storing the log on the same disk as the data:

data and log together on same device

All three bullets under "Disadvantages" are true.

data and log on separate segments on same device

Second and third bullets under "Disadvantages" are true.

data and log on separate devices, same disk

For "Disadvantages", third bullet is true, and second bullet is true if the log partition fails.

Where to Place the Log (cont.)

With larger tables, and where recoverability, uptime, and performance are critical, you have the following options:

- Place the log on a separate disk from the data
- If you expect a lot of activity, place the log on a disk that is (otherwise) not busy
- Place the log on its own segment so you can use the threshold manager to monitor its use
- Mirror the log device for full recoverability

threshold manager

See modules on Backing Up Databases & Logs and on Monitoring & Troubleshooting SQL Server for more details.

placing data and log on same device

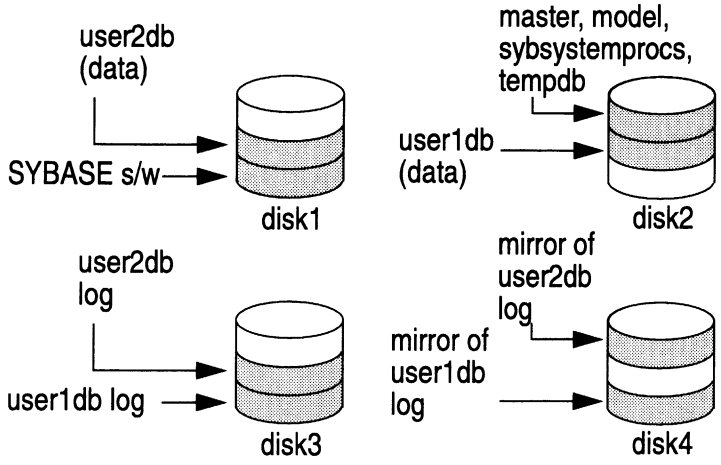
Note that both `create database` and `alter database` require the use of `with override` if you try to put the log on a separate segment but on the same device. For example:

```
create database newdb on dev1=3
   log on dev1=2
   with override
```

SYBASE

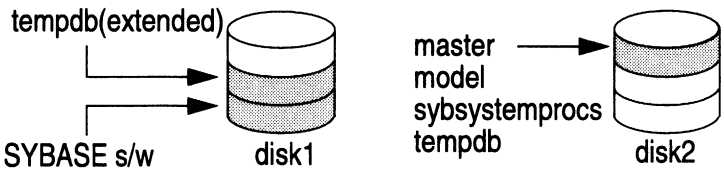
Where to Place the Log (cont.)

- Log on separate, relatively inactive disk, mirrored:

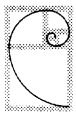


Where to Place *tempdb*

- The *tempdb* database is automatically installed on the master device
- If you need more space, you can expand *tempdb* onto other devices:



- If you expect tempdb to be quite active, place it on a disk that is (otherwise) not busy

SYBASE 

Where to Place Backups, *sybsecurity*

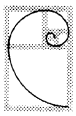
- Backups are generally placed on disk files or tapes
- The *sybsecurity* database contains the *sysaudits* table
- If you expect a lot of auditing activity, place *sybsecurity*
  - on a disk that is not busy, and
  - on its own segment (so you can use the threshold manager to monitor its use)

Backups See module on Backing Up Databases & Logs.

sybsecurity See module on Auditing.

Decisions You Need to Make

- ✓ What kinds of physical devices to use
- How much disk space you will need for:
  - ✓ data
  - ✓ indexes
  - ✓ logs (optional)
  - ✓ *tempdb*
  - ✓ backups
  - ✓ *sybsecurity/sysaudits*
- ✓ Where to place each of these
- Whether to mirror devices
- Whether to use segments

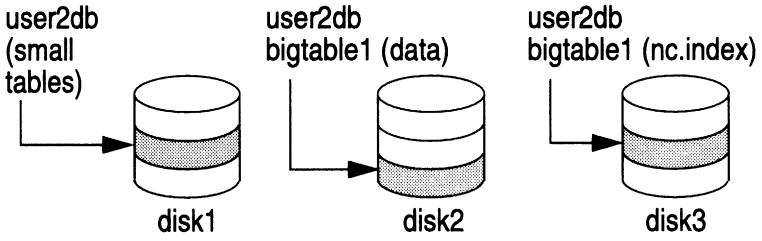
SYBASE 

Deciding Whether to Mirror Devices

- Mirrors are used to
  - prevent downtime due to disk failure
  - ensure full, non-stop recovery
- Entire *devices* are mirrored, not databases
- Mirror most valuable and most vulnerable
  - master device
  - log devices
  - active data devices
- Disk mirroring is highly recommended

Deciding Whether to Use Segments

- If you have large, highly-used tables, use segments to spread the load across multiple disks
  - Place these tables or their indexes on different disks from other tables and indexes in the same database

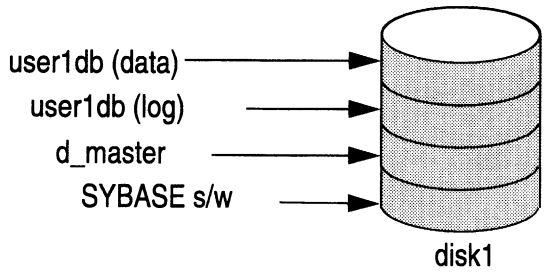


- Note that clustered indexes are inseparable from their tables

SYBASE

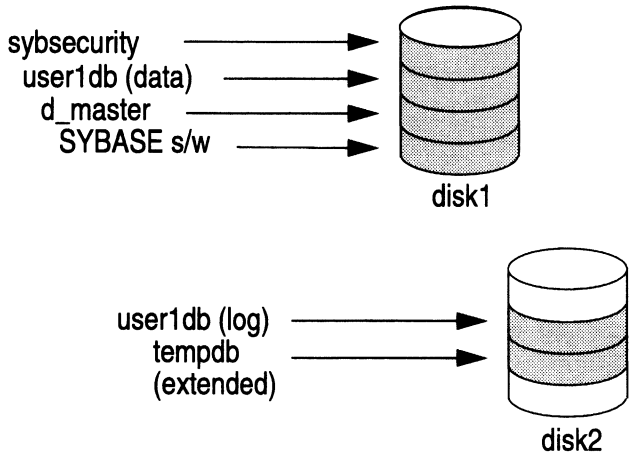
Putting it All Together

- In production, your goal is to balance the load of disk access across available disks
- In development (while experimenting, moving, sizing) you may choose to put everything on a single disk
- Example 1: Development environment



Putting it All Together (cont.)

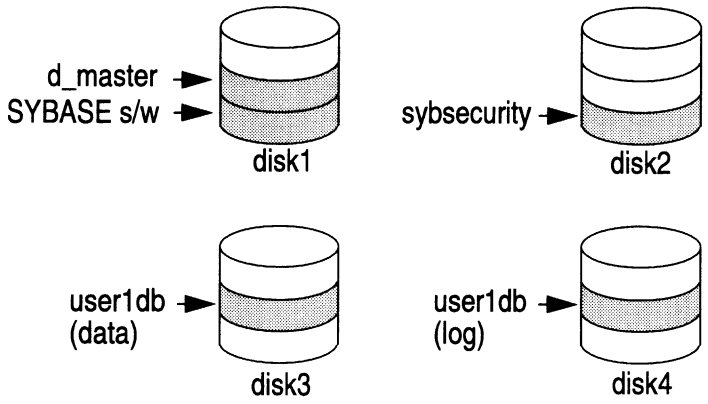
- Example 2: Two-disk production system



SYBASE

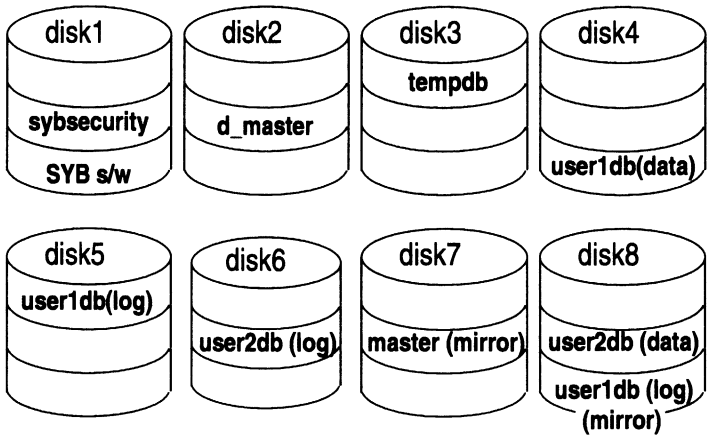
Putting it All Together (cont.)

- Example 3: Larger, four-disk production system



Putting it All Together (cont.)

- Example 4: Very large, multi-disk production system



- Assumption: user2db has low activity

Summary of System Tables

System Table	Displayed By	Updated By
sysdatabases (master)	sp_helpdb	create database drop database sp_changedbowner sp_dboption
sysusages (master)	sp_helpdb *database*	create database alter database drop database sp_logdevice sp_addsegment sp_extendsegment sp_dropsegment
sysdevices (master)	sp_helpdevice	sp_addumpdevice disk init
sysindexes (all databases)	sp_help sp_helpindex	create table create index
syssegments (all databases)	sp_helpsegment	sp_addsegment sp_dropsegment

Summary

- When sizing a database, consider mainly tables, indexes, and the transaction log
- Databases are easy to expand (using `alter database`), impossible to shrink
- For maximum uptime and full recovery, place logs on separate disks and mirror log devices
- For better performance, spread the work evenly across the disks you have, minimizing contention
- For a database that spans at least three disks, use segments to control placement of large, heavily-used objects

where to put active databases

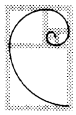
- Put active databases on otherwise inactive disks. (There is no performance advantage to putting active logs, etc., on separate partitions on the same physical disk.)

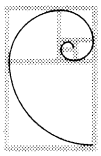
active disks

- Some obviously active disks: "swap" partition; partition where log is placed, partition handling print spooler, news reader, postmaster; partition handling remote mounts.

inactive disks

- Some obviously inactive disks: the disk with just the master device; the disk archiving old versions of software; the disk archiving old projects.





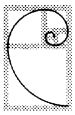
SYBASE®

Module 5

Controlling Access

Objectives

- Describe SQL Server roles and associated authorizations
- Add and manage SQL Server logins and database users
- Grant and manage SQL Server roles
- Grant and revoke user privileges

SYBASE 

Database Object Owners

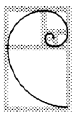
- A user who creates a database object is its owner
  - This is typically the Database Owner, but dbo may grant create permissions to database users
- Database Object Owners:
  - Have privileges over their own objects
  - Can grant permissions on their objects to other users, including the dbo
  - Cannot grant permissions that modify object's schema, such as `create index, alter table, drop object`
- Object ownership cannot be transferred

object permissions

Users (including dbo) do not have permissions over objects they do not own unless these permissions (e.g. select, insert, delete, execute) have been explicitly granted. We will see how this is done later in the module.

Database Owners (dbo)

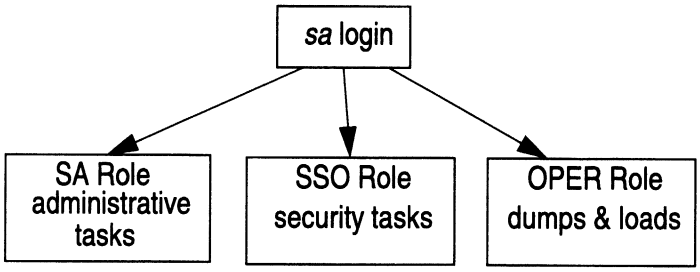
- The Database Owner is either
  - the creator of a database, or
  - someone to whom database ownership has been transferred
- Database Owners:
  - Add or drop database users
  - Can back up or load the database
  - Can execute `checkpoint` and `dbcc` in their database
  - Can execute `setuser` and become any database user
  - Via `setuser`, have full privileges on objects in their database
  - Can grant other users permissions on objects and commands in their database

SYBASE 

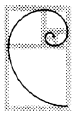
The "*sa*" Login

- When SQL Server is first installed, the *sa* login:
  - Has been assigned three special roles, and can execute all SQL commands
  - Is the owner of the *master* database
  - Is treated as database owner in every database
  - Has access to all databases and objects
- Password is initially null, but should be changed
  - Once changed, cannot be reset to null

The *sa* Login & the Three Special Roles



- At installation, *sa* has the three administrative roles and is authorized to do everything
- You can keep it that way, or you can grant these roles to one or more logins and then lock the *sa* login (or revoke its roles)

SYBASE 

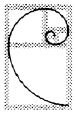
System Administrators (SA role)

- System Administrators can:
  - Manage disk storage
  - Drop, modify, and lock logins
  - Grant/revoke SA role
  - Create user databases; grant ownership of them
  - Grant certain permissions to SQL Server users
  - Use certain tools to diagnose system problems
  - Fine-tune SQL Server by changing configurable parameters
  - Shut down SQL Server and processes
  - Monitor recovery
- System Administrators become Database Owners in any database

System Security Officers (SSO role)

- System Security Officers can:
  - Create logins
  - Lock logins
  - Administer passwords
  - Grant and revoke SSO and OPER roles
  - Manage the audit system

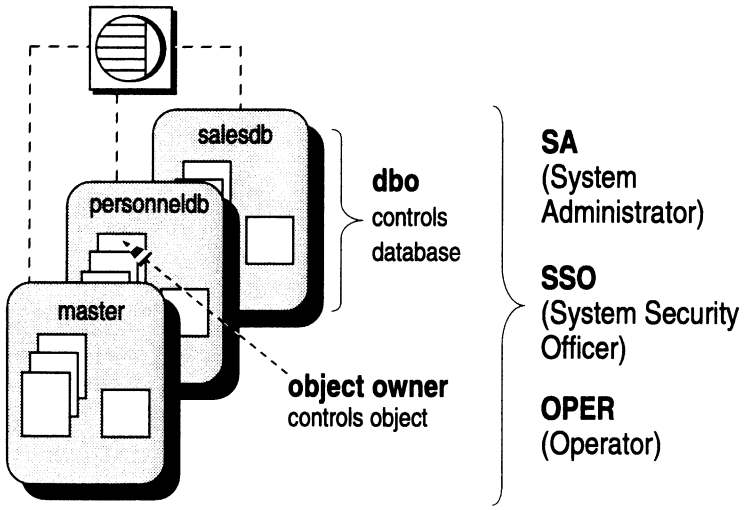
the audit system See module on Auditing.

SYBASE 

Operators (OPER role)

- Operators can back up and load *all* databases and transaction logs
- The Database Owner can perform these for his/her own database(s)

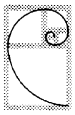
SQL Server Roles & Special Users



SA, SSO, OPER roles — Logins with these roles can perform certain server-wide tasks.

dbo — Users who are database owners have authority over the databases they own.

object owners — Users who own objects (such as tables, stored procedures, views) have authority over those objects



Lab 5a: Roles & Special Users

1. Which roles are associated with the following tasks? Assume the defaults, i.e., no permissions explicitly granted. Where there is more than one correct answer, fill in all that apply.

   Possible answers: SA, SSO, OPER, dbo, None of the above

   Manage the audit system _____

   Add new server logins _____

   Back up a database _____

   Create a table _____

   Run dbcc in a database _____

   Create a database _____

   Install SQL Server _____

   Drop database users _____

   Change login passwords _____

   Checkpoint a database _____

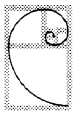
   Access a table _____

   Load a database _____

   Add a dump device _____

SQL Server System Security

- Historically, SQL Server has had a layered, object-based approach to system security
Access to:	**Controlled by:**
server, database objects, commands	adding/dropping logins, users granting/revoking permissions
- You can define groups of users and grant permissions on objects or commands to these groups
- With System 10, system-defined roles have been added to the picture
  - User-defined roles are planned for a future release

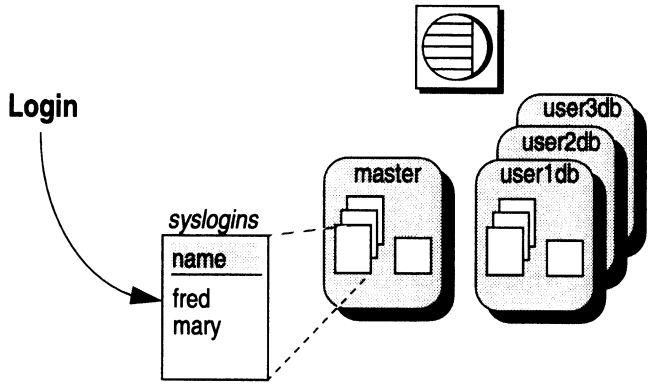
SYBASE 

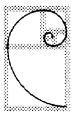
System Security: Topics

- Logins
- Users
- Groups
- Roles
- Command permissions
- Object permissions
- Auditing

SQL Server Logins

- In order to connect to SQL Server, you need a login
- Logins are listed in the *syslogins* table in *master*



SYBASE 

Adding Logins

- System Security Officers can add logins using `sp_addlogin`
- Full syntax:

```
sp_addlogin login_name, passwd [, defaultdb
  [, deflanguage [, fullname]]]
```

- Samples:

```
sp_addlogin claire, bleurouge, public_db,
  french, "Claire Barr"
sp_addlogin robby, playball, education
```

- Notes
  - Passwords: at least 6 bytes; null *not* allowed
  - If no default database is specified, default is *master*--not a good idea!

defaultdb

- default database, i.e., where the user will be connected when he/she logs into SQL Server
- will work only if the login is a user in the default database!

deflanguage

- the language assigned when a user logs in to SQL Server
- if unspecified, SQL Server's default language, defined by configuration variable "default language", is used

fullname

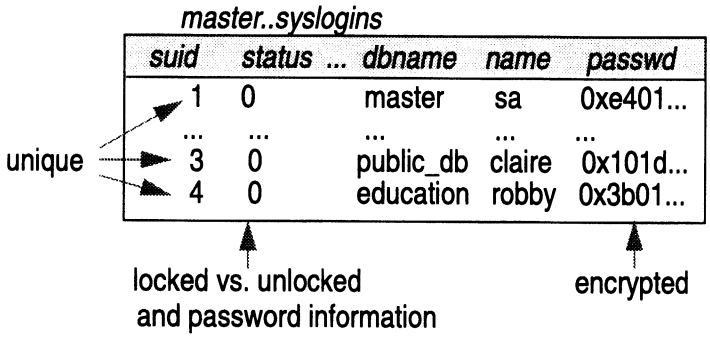
- the full name of the user
- can be used for documentation and identification purposes

optional arguments

- to leave out an optional argument and specify subsequent ones, use null (no quotes) or @ syntax
- Example:
  sp_addlogin claire, bleurouge, null, french
  sp_addlogin claire, bleurouge, @fullname="Claire Barr"

The *syslogins* Table

- When you add a login, a row is added to *syslogins* in *master*



- To view *syslogins*, use `select`

Sample output:

```
1> select suid, status, dbname, name, password, language, pwdate
2> from syslogins
3> go
 suid   status dbname                          name
         password
         language                        pwdate
 ------ ------ ------------------------------ ------------------------------
        ------------------------------------------------------------
        ---------------------------- --------------------------

      1      1 master                         sa
        0x20015660a70433ac4b17b2133f0eae26fa88a05fb83874f11b19e2edb620
        NULL                                  Mar  6 1993  3:08AM
      2      0 master                         probe
        0x7d015d0df5b787046ced4c70fca82d43b38686a3f16a4e5ebc9a34d1f473
        NULL                                  Apr  1 1993  3:22PM
      3      0 master                         mpiller
        0x260196e734515f336903e58afd4583a0a89e2a236b4038a48001b75a5568
        NULL                                  Apr  6 1993  2:33PM
      4      2 pubs2                          robby
        0x4d01e3515a4d6d43d2a2e5d10e53ae9427b49afc8a5e412fa2835d56344a
        us_english                            Apr 28 1993  4:06PM

(4 rows affected)
```

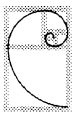
Status:

0 - Nothing special

1 - Short password (less than 6 bytes, set during upgrade on those passwords that are < 6 bytes, and also for the 'sa' and 'probe' logins during install, since the passwords are null).

2 - Login is locked

4 - Password is expired

SYBASE 

Login-Related Procedures

- SAs:
  - sp_modifylogin *login_name, option, value*
  - sp_droplogin *login_name*
- SSOs:
  - sp_addlogin *login_name, password, ...*
  - sp_password *caller_passwd,new_passwd [,login_name]*
  - sp_configure password, *n*
- SAs or SSOs:
  - sp_locklogin [*login_name*, "{lock | unlock"}]

What is the difference between locking and dropping a login?

What if all SSOs leave town, or their passwords are lost or forgotten?

modifying logins

- possible options: *defdb, language*, and *fullname*

dropping logins

- cannot drop logins that are users in a database (and cannot drop users that own objects!)

setting passwords

- passwords are encrypted, and cannot be retrieved
- to set password, use sp_password
- "caller_passwd" refers to password of login executing sp_password

setting password expiration interval

- sets number of days that passwords remain in effect after they are changed
- users are warned before passwords expire;
- if a user's password expires, he/she can log in but cannot execute any commands other than sp_password

locking vs. dropping

- a locked login remains listed in *syslogins*; user cannot access SQL Server until the login is unlocked
- a dropped login is dropped from *syslogins* and would have to be added using sp_addlogin

what if all SSOs leave town?

- option to `startserver` generates new password

Niet meer!

System Security: Topics

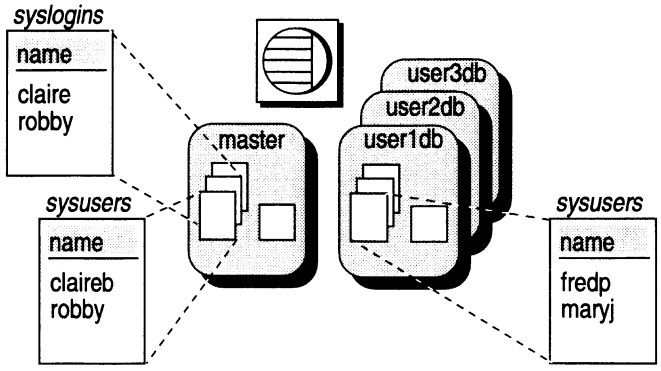
✓Logins

- Users
- Groups
- Roles
- Command permissions
- Object permissions
- Auditing

SYBASE

Database Users

- In order to access a database, you need to be listed as a user in the database
- Users are listed in the *sysusers* table in each database



exception

If *guest* is listed in *sysusers*, logins that are not listed in *sysusers* can access the database. Details later in this section.

Adding Users

- Full syntax:

```
sp_adduser login_name [, name_in_db
  [, grpname]]
```

- Samples:

```
sp_adduser claire, claireb
sp_adduser robby
```

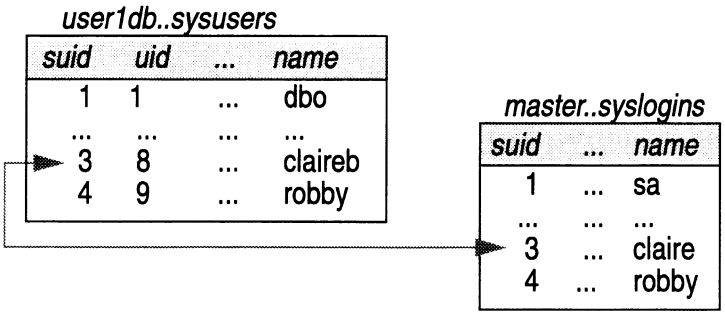
- Only Database Owners can add users
  - System Administrators automatically become dbos in each database they access, thus can add users

grpname groupname - We will be discussing groups later in this module.

SYBASE

The *sysusers* Table

- When you add a user, a row is added to *sysusers* in that database



- To view *sysusers*, use `select * from sysusers`
- To display a list of database users with their login names and default databases, use `sp_helpuser`

suid server user ID, copied from *syslogins*. Suid 1 is *sa*.

uid user ID, unique in this database. User ID 1 is *dbo*.

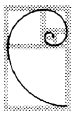
gid group ID, if any.

sp_helpuser sample output:

```
Users_name     ID_in_db     Group_name      Login_name        Default_db
------------------------------------------------------------------------
dbo                   1     public          sa                master
...
claireb               8     public          claire            public_db
robby                 9     public          robby             education
...
```

Guest Users

- Adding a user named "guest" in a database enables any SQL Server login to access the database as a guest
- To add a guest user, use `sp_adduser guest`
  - The *uid* for guest is 2
- When SQL Server is installed, *sysusers* in *master* and *tempdb* contain an entry for guests
  - Guest users cannot be dropped from these databases
- Do not add a *login* called "guest"

SYBASE 

Aliases

- Can treat more than one login as the same database user, giving all of them the same privileges
  - Can map several users to *dbo*, or to another user name
  - If activities of this alias are audited, the "real" identity can be recorded as well
- To create an alias, use `sp_addalias`

  ```
  sp_addalias loginame, name_in_db
  ```
- Adds a row to *sysalternates* in the database

sp_addalias

- login must not already be a user in the database
- *name_in_db* must exist in *sysusers* and *master..syslogins*

Database Access

- When you try to access a database, SQL Server:
  - checks sysusers for your suid; if no match, then...
  - checks sysalternates for your suid; if no match, then...
  - checks sysusers for "guest"
- If all three fail, SQL Server denies access

SYBASE

Revoking Access to SQL Server or Databases

- `sp_droplogin` *`login_name`*
- `sp_locklogin [`*`login_name`*`, "{lock | unlock}"]`
- `sp_dropuser` *`name_in_db`*
- `sp_dropalias` *`login_name`*

sp_droplogin
- discussed previously

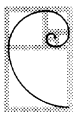
sp_locklogin
- discussed previously

sp_dropuser
- cannot drop a user who owns objects in the database
- if there are aliases to that user, the aliases are cancelled

sp_dropalias
- login aliased must be added as a user in order to be able to access the database

System Security: Topics

- ✓ Logins
- ✓ Users
- Groups
- Roles
- Command permissions
- Object permissions
- Auditing

SYBASE 

Groups

- Enable you to provide a collective name to a group of users
- Are a convenient way to organize users with common permissions
- Example

```
sp_addgroup engineering
```

Then

```
sp_adduser bob, bob, engineering
```

or

```
sp_changegroup engineering, bob
```

- The group *public* is always present, and everyone is automatically in it
- Limitation: users can belong to only one group besides *public*

groups and user ids (uid)

- "public" has a uid of 0
- other groups have uid's > 16383
- roles (sa_role, etc) appear as groups in *sysusers* and have uid's > 16383 as well

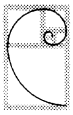
groups and permissions

- you can grant or revoke permissions to the group, affecting all the users in that group

Displaying Groups

- Use `sp_helpgroup` to display the groups in a database
- Syntax:

  `sp_helpgroup [grpname]`
  - Without a parameter, returns all groups in the database
  - With a parameter, returns users in that group
- The *sysusers* table has a row for each group
  - Roles appear as groups in *sysusers* of all databases but work quite differently

SYBASE 

Useful Functions & System Procedures

- Built-in functions:

suser_id( )	db_id( )	user_id( )
suser_name( )	db_name( )	user_name( )

- Examples
  - `select suser_id( )`
  - `select suser_id ("fred")`
- System procedures
  - `sp_helpuser`
  - `sp_helpgroup`

Lab 5b: Logins, Users & Groups

In this lab, we ask you to add three logins to SQL Server and add these logins as users in your database. We also ask you to create a group and add certain users to that group.

1. Add three logins to SQL Server. Call them *tomN*, *dickN*, and *harryN* (where N is your user number), and make your database their default database. Give them all "friday" as a password.

2. Add these logins as users in your database.

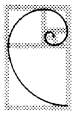
3. *harryN* has forgotten his password and needs help. If you were a System Security Officer, what could you do? Write down the command you would use.

   sp_password <sa-pw>, saturday, harry6q

4. Create a group called *programmers* and add users *tomN*, *dickN*, and *harryN* to that group. Execute `sp_helpgroup` and `sp_helpgroup programmers`, and note the difference in output.

System Security: Topics

- ✓ Logins
- ✓ Users
- ✓ Groups
- Roles
- Command permissions
- Object permissions
- Auditing

SYBASE 

Roles

- When SQL Server is first installed, the *sa* login has SA, SSO, and OPER roles
- To grant these roles to other logins, use `sp_role`
- Full syntax:

```
sp_role "{grant | revoke}",
  "{sa_role | sso_role | oper_role}",
  login_name
```

- Examples:

```
sp_role "grant", "sa_role", robby
```

- SSOs grant/revoke SSO and OPER roles;
  SAs grant/revoke SA role

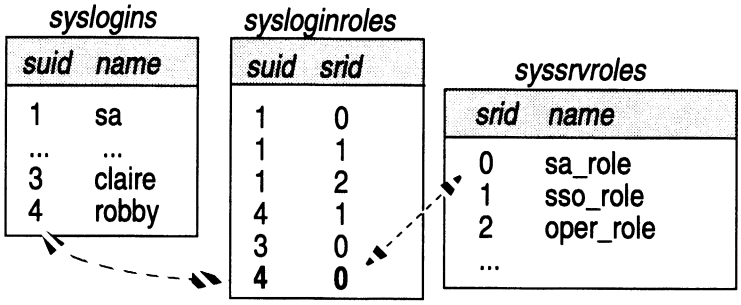
revoking roles

The last remaining unlocked logins with SA or SSO roles cannot have the role revoked.

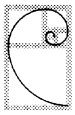
Also, you cannot revoke a role from a login that is currently logged on.

What `sp_role` Does

- When you execute `sp_role`, a row gets added or deleted in *master..sysloginroles*
- Example: sp_role "grant", "sa_role", robby



- *sysloginroles* contains configured role assignments
  - These take effect at next login

SYBASE 

Displaying Information About Logins

- `sp_displaylogin` displays information about a specific login, including what roles have been granted

```
1> sp_displaylogin robby
2> go
Suid: 4
Loginame: robby
Fullname: Robert Morris
Configured Authorization: sa_role sso_role
Locked: NO
Date of Last Password Change: Apr 30 1993  1:30PM
(return status = 0)
```

- Roles listed are those currently configured

displaying roles

- All logins can display their own information
- SAs and SSOs can display information for other logins

Determining What Roles Are Active

- Use the built-in function show_role( ) to determine which roles are currently enabled for your login

```
select show_role()
```

- Within a stored procedure, use the built-in function `proc_role( )` to check role assignments of the login calling the procedure

```
create proc ...
as
if (proc_role("sso_role") != 1)
return(1)
```

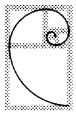
- You can use this mechanism to ensure that a stored procedure is executed only by logins which have a certain role or roles

show_role( )

This function does not take parameters.

proc_role( )

- returns 1 when login executing this procedure has the named role
- returns 0 when login does *not* have the named role.

SYBASE 

Disabling/Enabling Roles

- When you log in, the roles that have been granted to your login, if any, are automatically enabled
- To disable a role, use

```
set role
  "{sa_role | sso_role | oper_role}" off
```

- Example

```
set role "sa_role" off
```

  This will take effect immediately and remain in effect for the duration of the session, unless re-enabled

- To re-enable a role, use `set role ... on`
- Why might you disable a role?

Role Management

- We recommend granting SA, SSO, and OPER roles to specific logins, and then locking the *sa* login
  - This increases user accountability, since you can audit activities of specific logins
- To lock a login, use `sp_locklogin`

```
sp_locklogin [login_name, "{lock | unlock}"]
```

  With no parameters, returns list of locked logins
- SQL Server will not allow you to lock the last unlocked login with the SA role, nor the last unlocked login with the SSO role
  - `sp_droplogin`, `sp_locklogin` and `sp_role` each have checks to ensure this

advantages of proposed approach

- not necessary to give out *sa* password for common administrative tasks
- spread out responsibilities, separating out those that are security-sensitive
- more flexible, as roles can be granted to more than one login, and a login can have more than one role
- can audit activities of any login, including logins that assume SA, SSO, and OPER roles

Lab 5c: Roles

In the last lab, you added three logins to SQL Server: tomN, dickN, and harryN. You also added these logins as users in your database, and to the group programmers.

In this lab, we ask you to grant certain roles to certain logins and experiment to see what tasks these logins can perform as a result. We ask you to add a login called lauraN. See next page for syntax of useful commands for the lab.

The first optional exercise asks you to write a short procedure that only System Administrators should be able to execute. Then we ask you to think about the situation in your business and think about what kind of site security policy you will implement.

1. Grant SA and OPER roles to *tomN*. Grant SSO role to *dickN*. Don't grant a role to *harryN*. Use `sp_displaylogin` to check role configuration. When will these take effect?

2. Log in as *tomN* and check what roles are active for that login. Try to add the new login *lauraN*, with your database as the default. Were you successful? Why or why not?

3. Log in as *dickN*, check what roles are active for that login, and add a new login, *lauraN*. Make your database her default database. (We will add her as a user in a later lab.)

4. As *dickN*, disable the SSO role (Hint: use `set`) and try to add a new login, *bobN*. What happens?

Optional:

5. Using your user name to log into SQL Server, create a stored procedure in your database that only logins with the SSO role can execute. The procedure should print "Success!" if successful, or exit with an error message if an login that does not have the SSO role attempts to execute it. Test this out , then disable the SSO role for your login and verify that you can no longer run the procedure. Then enable SSO role again.

6. Think about the business requirements of your company and define an appropriate site security policy. Include the following elements:
   a) Will you have an unlocked *sa* login with all roles enabled?
   b) How many logins (if any) will have SA, SSO, and OPER roles?
   c) What stored procedures will you require special authorization for?
   d) What groups will you form, and which users will belong to these groups?

Useful Commands for the Lab

Displaying configured roles:

```
select show_role()
```

Checking which roles are active:

```
sp_displaylogin [ login_name ]
```

Granting/revoking roles:

```
sp_role "{grant | revoke}",
  "{sa_role | sso_role | oper_role}", login_name
```

Enabling and Disabling roles:

```
set role "{sa_role | sso_role | oper_role}" on|off
```

System Security: Topics

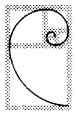
✓Logins

✓Users

✓Groups

✓Roles

- Command permissions
- Object permissions
- Auditing

SYBASE 

Granting and Revoking User Privileges

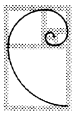
- System Administrators can grant certain privileges to other users
  - `create database`
- Database Owners control permission to execute certain database commands
  - `create procedure`
  - `create default, rule, table, view`
- Object owners control access to their own objects
  - `select, insert, delete, update, execute`

Granting Command Privileges

- Use `grant` to give command permissions

```
grant
  {all [privileges] | command_list}
  to {public | name_list | role_name}
```

- Examples:
  - `grant create rule to fred`
  - `grant create rule, create table to engineering`
  - `grant create database to mary, john` (SA only)
- Database command permissions that cannot be given away:
  - `dbcc, dump database, dump tran, load database, load transaction, setuser`

SYBASE 

Revoking Command Privileges

- Use `revoke` to remove command permissions

```
revoke {all [privileges] | command_list}
  from {public | name_list | role_name}
```

- Examples

```
revoke create table from public
revoke create rule from fred
revoke create database from mary, john
```

  - Only the sa can `revoke create database`

Granting Object Permissions

- Use grant to give permissions on specific objects

```
grant {all [privileges] | permission_list}
  on {table_name [(column_list)]
     | view_name [(column_list)]
     | stored_procedure_name}
  to {public | name_list | role_name}
  [with grant option]
```

- Samples:

```
grant insert, delete on titles to mary, sales
grant update on titles (price, advance)
  to public
grant execute on new_proc to sa_role
```

- Permissions that cannot be granted:

```
create index, create trigger, alter table,
  drop table, truncate table, update
  statistics
```

permission_list • select, insert, delete, references, update

with grant option • See later page.

SYBASE

Revoking Object Permissions

- Use `revoke` to remove permissions on specific objects

```
revoke [grant option for]
  {all [privileges] | permission_list}
  on {table_name [(column_list)]
     | view_name [(column_list)]
     | stored_procedure_name}
  from {public | name_list | role_name}
  [cascade]
```

- Samples:

```
revoke insert, delete on titles
  from mary, sales
revoke update on titles (price, advance)
  from public
revoke execute on new_proc from sa_role
```

revoke grant option for...
revoke...cascade

- See upcoming pages.

grant...with grant option

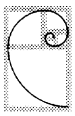
- Allows specified user(s) to grant specified privileges to other users

```
grant select on mytable to philip
  with grant option
```

- As *philip*:

```
grant select on mytable to mary
```

- You can `grant...with grant option` only to individual users, not to *public*, groups, or roles
- Manage this carefully, or you can lose control over object permissions!

SYBASE 

revoke grant option...cascade

Given the situation on the previous foil:

1) `revoke select on mytable from philip`
   - grant and select authority revoked from philip, and select (and grant authority, had this been granted) revoked from mary

2) `revoke select on mytable from philip cascade`
   - same as above

3) `revoke grant option for select on mytable from philip cascade`
   - grant authority revoked from *philip*, select (and grant authority, had this been granted) revoked from *mary*

4) `revoke grant option for select on mytable from philip`
   - Command fails, as *philip* has granted permissions to *mary*

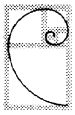
Sequencing Permissions Commands

- Chronology determines resulting permission
- To exclude one or more users:

```
grant create table to public
revoke create table from mary

grant select on payroll to clerks
revoke select on payroll from nancy, james
```

- To protect one (or more) column(s):

```
grant select on payroll to public
revoke select on payroll(salary)
  from public
```

SYBASE 

Create Schema Authorization

- The `create schema` command creates a collection of objects owned by a single user
  - It can include permissions granted on those objects
- Sample:

```
create schema authorization anna
  create table tableA (col1 int, col2 int)
  create view v1 as select col1 from tableA
  grant select on v1 to public
```

- Purpose: to bundle creates and permissions of related tables in one transaction
- If any statements within the schema fails, the entire command is rolled back

syntax

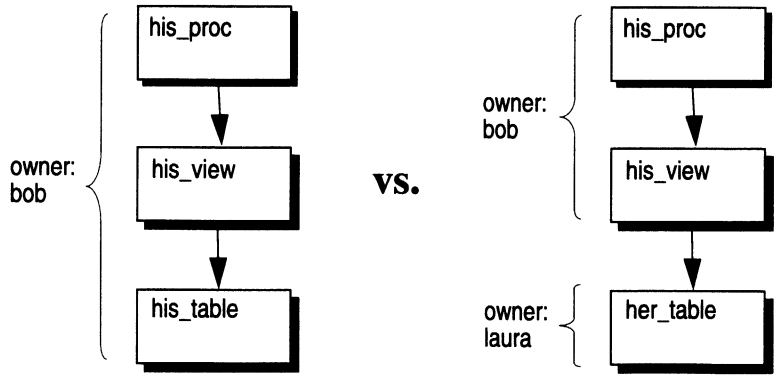
```
create schema authorization
  authorization_name
create_object_statement
  [ create_object_statement ... ]
[ permission_statement ]
```

authorization_name is the database user name of the user executing the command.

how create schema authorization works

The create schema authorization command creates the objects contained within it but leaves no record of itself.

Ownership Chain: 2 Cases

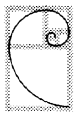


how the ownership chain works

- When you access an object that depends on another object (which may depend on another, etc), permissions are checked on the first object only, unless ownership changes.

how to control access to an object

- You can control access to an object by granting permissions only on an object that depends on it.
- For example, deny access to a table but grant access to a view or stored procedure based on that table.

SYBASE 

Displaying Permissions

- Granting command and object privileges adds a row in *sysprotects*
- To display permissions for a given object use `sp_helprotect`

```
sp_helprotect titles
```

- To display permissions for a given user, use `sp_helprotect` *`user_name`*

```
sp_helprotect laura
```

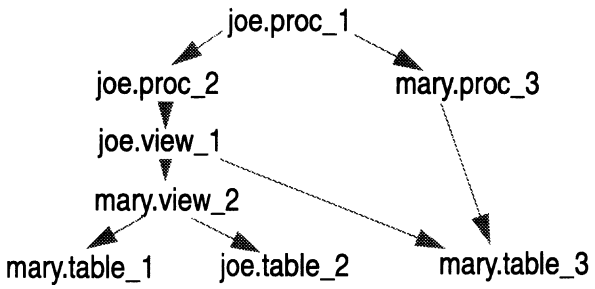
Permissions on System Procedures

- System Administrators control access to system procedures and system tables in *master*
- Characteristics of system procedures:
  - Reside in *sybsystemprocs* but can be run from any database
  - Permissions set in *sybsystemprocs..sysprotects* by System Administrators
- The *installmaster* script grants permissions to *public* for certain system procedures
  - System Administrators can choose to revoke these
- To create a new system procedure that everyone can use:
  - Create it in *sybsystemprocs*; have it begin with "sp_",
    `grant execute to public`

SYBASE

Permission Strategy

- Worst approach: Designed by committee



- Best approach: Keep ownership the same

Summary

- In developing a site security policy, consider the following:
  - Whether to have an all-powerful *sa*, or use roles
  - Which users (if any) should have special privileges
  - Which procedures and objects need protection
- Once these are established, you can decide whether to audit certain activities or certain logins

auditing See module on Auditing.

Lab 5d: Grant & Revoke Privileges

As of the last lab, you had created four logins: tomN, dickN, harryN, and lauraN. tomN has SA and OPER roles, and dickN has SSO role.

In this lab, we ask you to create a table and grant certain permissions on it to specified users. We also ask you to display permissions for specific users and groups.

1. As *tomN* (who is also SA/dbo), create a two-column table in your database and insert a few rows.

2. Create a stored procedure that prints a brief message to the screen and then selects all rows from the table.

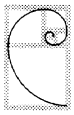
3. Add *lauraN* as a user in your database. Then grant and revoke permissions so that:
   a) all users except *lauraN* can create stored procedures in the database
   b) *dickN* can select, insert, and update rows in the table you created, but not delete
   c) *harryN* can execute your procedure and can grant this permission to others
   d) *lauraN* can select just the first column of the table
   e) all users in the *programmers* group can create tables

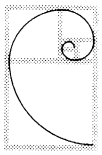
4. Display protections for the table and stored procedure, and display permissions for specific users and groups.

5. Can *harryN* execute your procedure? As he does not have permission to select from the table, how do you explain this?

6. What command(s) would you use to ensure that only System Security Officers could run `sp_helpuser`? (Don't do this, however.)

Revoke from public
grant to SSO





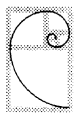
SYBASE®

Module 6

Configuring SQL Server

Objectives

- Configure SQL Server appropriately
  - memory
  - user connections
  - devices
  - procedure cache
  - etc.

SYBASE 

What Configuration Is

- Configuration options control various aspects of SQL Server's memory allocation and performance
- Sample configurable options
  - memory
  - user connections
  - devices
  - procedure cache
  - open databases
  - recovery interval
  - remote access
  - default language

Why Configuration Matters

- Configuration values manage use of resources and affect performance
- Default configuration values, shipped with the product, are minimal and will allow SQL Server to start
  - System Administrators can "reconfigure" (reset) most of these
  - System Security Officers control a few of these

SSO

Can affect *password expiration interval, audit queue size, allow updates.*

SA

Can configure all the others, except for default character set id and default sortorder id, which are determined at installation

Displaying Configuration Values

- Use `sp_configure` to display current configuration valuesl

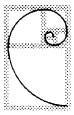
```
1> sp_configure
2> go
 name                            minimum     maximum     config_value run_value
 ------------------------------- ----------- ----------- ------------ -----------
 recovery interval                         1       32767            0           5
 allow updates                             0           1            0           0
 user connections                          5  2147483647            0          25
 memory                                 3850  2147483647         8192        8192
 open databases                            5  2147483647            0          10
 locks                                  5000  2147483647            0        5000
 open objects                            100  2147483647            0         500
 procedure cache                           1          99            0          20
 fill factor                               0         100            0           0
 time slice                               50        1000            0         100
 database size                             2       10000            0           2
 tape retention                            0         365            0           0
 recovery flags                            0           1            0           0
 nested triggers                           0           1            1           1
 devices                                   4         256            0          10
 remote access                             0           1            1           1
 remote logins                             0  2147483647            0          20
 remote sites                              0  2147483647            0          10
 remote connections                        0  2147483647            0          20
 pre-read packets                          0  2147483647            0           3
 upgrade version                           0  2147483647         1000        1000
 default sortorder id                      0         255           50          50
 default language                          0  2147483647            0           0
 language in cache                         3         100            3           3
 max online engines                        1          32            1           1
 min online engines                        1          32            1           1
 engine adjust interval                    1          32            0           0
 cpu flush                                 1  2147483647          200         200
 i/o flush                                 1  2147483647         1000        1000
 default character set id                  0         255            1           1
 stack size                            20480  2147483647            0       28672
 password expiration interval              0       32767            0           0
 audit queue size                          1       65535          100         100
 additional netmem                         0  2147483647            0           0
 default network packet size             512      524288            0         512
 maximum network packet size             512      524288            0         512
 extent i/o buffers(for sort)              0  2147483647            0           0
 identity burning set factor               1     9999999         5000        5000

(38 rows affected, return status = 0)
```

Displaying Configuration Values (cont.)

- to display the values for a particular configuration option, use `sp_configure` *option*
- Example:

```
1> sp_configure "remote access"
2> go
name          min  max  config_value run_value
-------------------------------------------
remote access   0   1            1         1
```

- All logins have permission to display these values

SYBASE 

Two Kinds of Configuration Options

- Dynamic options, which can be changed without restarting SQL Server
  - Some examples:
    recovery interval, default language, nested triggers, password expiration interval
- Non-dynamic options, which do not take effect until SQL Server is restarted
  - Some examples:
    user connections, memory, open databases, locks, procedure cache, remote logins, stack size, default network packet size

Setting & Installing Configuration Options

- Dynamic options
  - Set using `sp_configure`
    ```
    sp_configure "option", value
    ```
  - install using `reconfigure`
- Non-dynamic options
  - set using `sp_configure` (syntax as above)
  - install using `reconfigure`, then reboot SQL Server
- SAs set most options, SSOs set a few
- Only SAs can use `reconfigure`, and if necessary, reboot
  - If you supply values SQL Server finds unreasonable, you will need to `reconfigure with override`

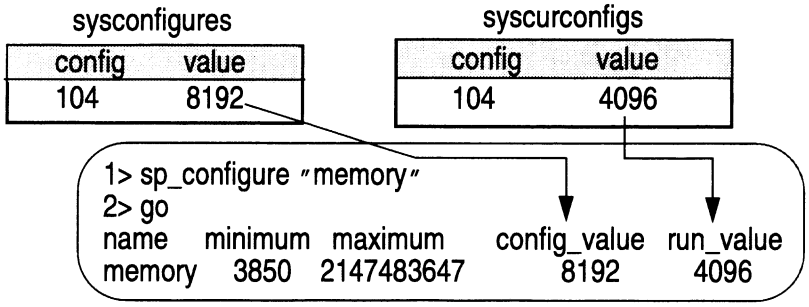
setting & installing: exceptions

default sortorder id and default character set id cannot be reconfigured without reinstalling SQL Server.

SYBASE

sysconfigures, syscurconfigs, sp_configure

- The system tables *syscurconfigs* and *sysconfigures* contain current and "future" configuration values, respectively
- sp_configure uses these tables to display both values



run_value = the actual, runtime value

config_value = value to take effect after reconfiguring

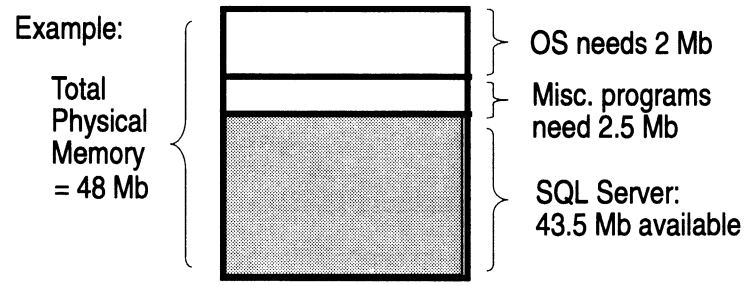
config_value

A config_value of zero (0) indicates the default for the current platform.

SYBASE Troubleshooting Guide Chapter 2, "SQL Server Configuration Issues".

How Much Memory to Configure

- Memory is the most important configuration option!
  - Not setting this correctly will affect performance dramatically
- To optimize the size of memory for your system:
  - Calculate memory required for OS and other system uses; subtract this from total available physical memory



if SQL Server has too much memory

- Server may not boot
- If it does boot, OS page fault will go up dramatically; may need to reconfigure OS to compensate

if SQL Server has too little memory

- Server may not boot
- If it does boot, SQL Server must access disk more frequently

In both cases, performance will suffer.

SYBASE

When Memory Is Allocated

- SQL Server requests memory from the operating system only at start-up time, asking for memory based on configuration value

```
1> sp_configure "memory", 8192
2> go
...
1> sp_configure "memory"
2> go
name    minimum     maximum     config_value  run_value
------- ----------- ----------- ------------- -----------
memory     3850  2147483647          8192        4096
```

*(Server will ask for 8192*2K)*

- SQL Server divides memory up depending on configured user connections, devices, etc.

Reconfiguring Memory

- To reconfigure memory:

```
sp_configure "memory", 8192
go
reconfigure
go
```

number of 2K pages

Then restart SQL Server

- Objective: to give SQL Server as much physical memory as possible without causing the OS to "swap out" or "page" virtual memory to disk
- Do not ask for more memory than physically available
- If SQL Server won't start, use `buildmaster` to recover
  - Unix: `buildmaster -r`
  - OpenVMS: `buildmaster /r`

OpenVMS

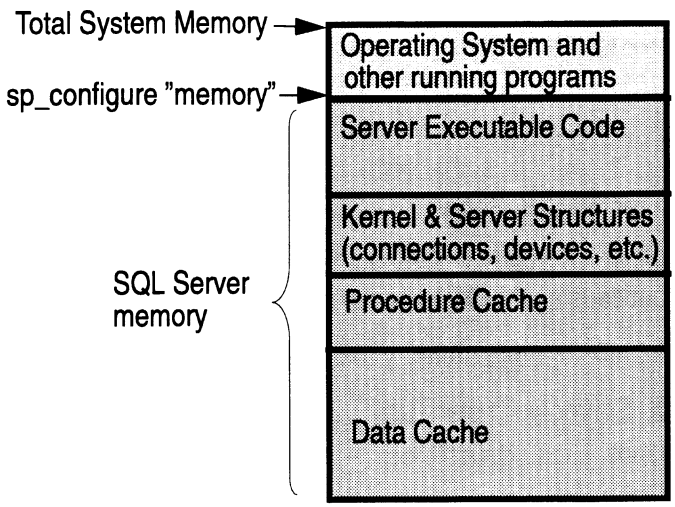
Be sure to follow Installation Guide instructions for changing quotas when changing memory size or user connections.

Unix:
`buildmaster -r`
OpenVMS:
`buildmaster /r`

If you use this option to `buildmaster`, the configuration block that contains system startup parameters is overwritten with the default values. Do not use this option unless SQL Server will not start!

See Utilities manual for details.

How Memory is Used



SQL Server executable code

The size of SQL Server executable code is fixed.

Configuration Options Related to Memory

- The following configuration options affect how SQL Server memory is used:

  Kernel & Server structures:

user connections	50 Kb per
devices	512 bytes per
open databases	7500 bytes per
open objects	315 bytes per
locks	72 bytes per
Procedure cache	configurable

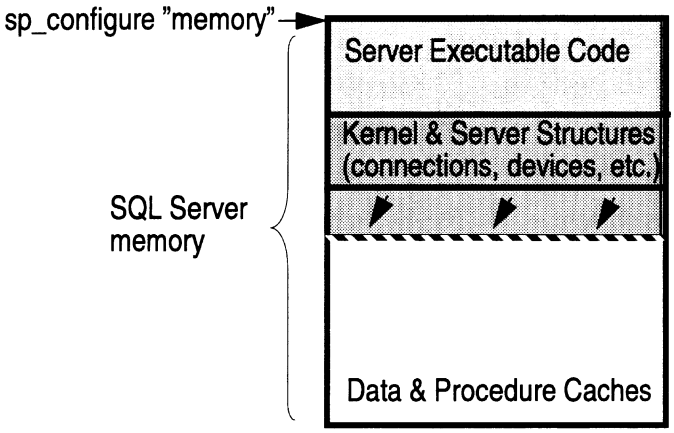
- All are configurable by System Administrators
  - Most significant are user connections and procedure cache

kernel & server structures Numbers given are approximations; actual values will vary.

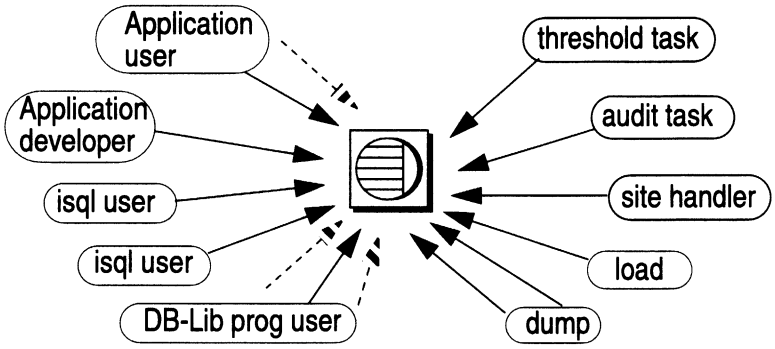
SYBASE

Kernel & Server Structures

- Grows at the expense of the data and procedure caches



How Many Connections Should You Allow?



- The "user connections" option sets the maximum number of user connections that can be connected to SQL Server at the same time
- The total number of connections may be limited by the OS

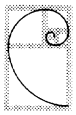
Unix

Unix limits the number of user connections since a process (e.g., SQL Server) can open only a certain number of i/o connections (file descriptors).

VMS

On VMS, there are a limited number of DECNET connects, which limits the number of user connections you can use.

Be sure to follow Installation Guide instructions for changing quotas when changing memory size or user connections.

SYBASE 

Setting User Connections

- Situation: There are currently 35 users. You want to add connections for 15 more users.
- First, calculate how much memory this will involve; allow 50K of memory for workspace per user connection
  - How much memory will be taken from the data and procedure caches?
- If you have enough memory available, then set the option, reconfigure, and restart SQL Server:

```
sp_configure "user connections", 50
go
reconfigure
go
```

Example

Adding 15 more connections would take 375 2K pages. (50K = 25 * 2K pages; 25 * 15 = 375.)

Note:

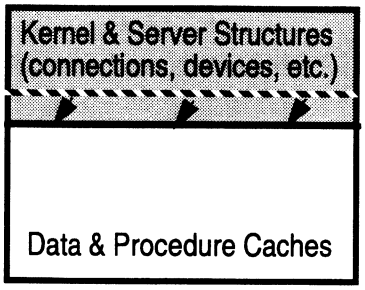
Calculate how much memory will be lost or gained before reconfiguring.

If you configure more than you have available, you will not be able to restart SQL Server.

See Troubleshooting Guide ("SQL Server Configuration Issues") for more information on estimating memory use for user connections and other options.

Devices

- This option refers to the number of database devices that SQL Server can use
- As with user connections, the higher this is set, the less memory left over for data and procedure caches

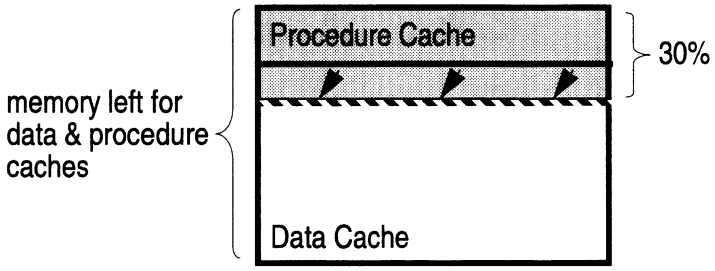


- Allow 512 bytes of memory per device

SYBASE

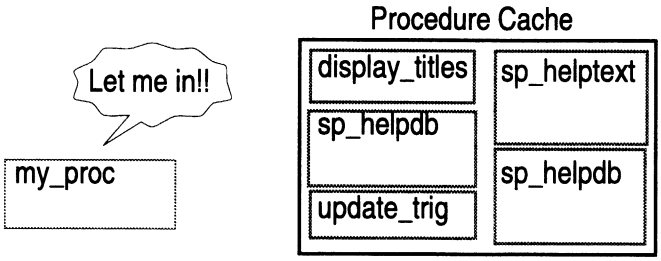
Setting Procedure Cache

- Procedure cache is set as a percentage of the available memory after the SQL Server executable and kernel and Server structures have been allocated
- Setting procedure cache higher decreases the memory available for data cache
- Example: `sp_configure "procedure cache", 30`

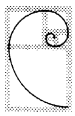


What Procedure Cache Is For

- Procedure cache holds the optimized query plans for all batches
  - Most significant: procedures currently in use, including triggers
  - If more than one user uses a procedure simultaneously, there will be multiple copies of the procedure in cache



- If you make the procedure cache too small, user processes may have to wait

SYBASE 

How Big Should Procedure Cache Be?

- Estimate how many concurrent procedures you will need space for
- Example
  - Assume:
    2 procedures: 10K and 25K
    Expect 30 users each, concurrently
  - How big will the procedure cache have to be?
- Or, use one of the following formulas as a starting point:
  - Max(# of concurrent users)*(size of largest plan)*1.25
  - (# of main procedures)*(avg. plan size)*1.10
- `dbcc memusage` displays query plan sizes of largest procedures in recent use

Reference

See SYBASE Troubleshooting Guide Chapter 2 for a discussion on Procedure Cache Sizing.

dbcc memusage

- `dbcc memusage` (undocumented) lists the query plan sizes of the 20 largest procedures currently in the procedure cache
- Output is in three parts:

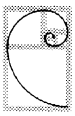
```
Memory usage:              Meg      2K Blks      Bytes
   Configured memory     16.0000      8192     16777216
   Code size              2.5686      1316      2693400
   ...
Buffer Cache, Top 20
   Dbid          Object Id          Index Id      2K Buffers
      6                  8                 0            1774
   ...
Procedure Cache, Top 20
   Database Id:  1; Object Id: 1456008218; ...; Type: stored
    procedure; ...; Size of plans:   0.109215 Mb...
```

dbcc memusage

This is an undocumented command and is not officially part of the product.

You must execute `dbcc traceon(3604)` before running `dbcc memusage`to send output to your screen. Reset using `dbcc traceoff(3604)`.

Note: Do not use dbcc memusage in a symmetric multiprocessing (SMP) environment as processes could be lost.

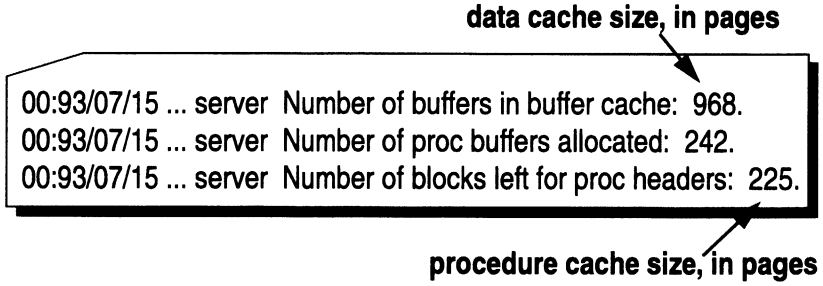
SYBASE 

Data Cache

- Data cache contains pages from recently accessed objects, typically:
  - *sysobjects*, *sysindexes*, *syslogs* for each database
  - Parts of indexes
  - Parts of actively-used tables
- You cannot explicitly set data cache: it is set for you, based on remaining memory
- Monitor cache effectiveness using `set statistics io on` to watch the ratio of physical and logical accesses
  - This is covered in the Performance & Tuning class

Finding Out Sizes of Procedure & Data Caches

- Read boot-up messages or error log to see how much data and procedure cache was allocated
- Sample extract from errorlog:



buffer cache

- indicates how many 2K buffers (pages) are used for data cache.

proc buffers

- a data structure used to manage compiled objects (stored procedures, triggers, rules, defaults, views) in the procedure cache.

The number of proc buffers represents the maximum number of compiled objects that can reside in the procedure cache at one time.

proc headers

- indicates number of 2K pages dedicated to procedure cache.

SYBASE

Configuration Example

- Assume total system memory of 64Mb, with a dedicated SQL Server
  - What configuration values would you use for a system for 60 connections?
- Sample calculations (approximations only)

Total Memory	64 Mb
Size of OS	2 Mb
Avail. for Server	62 Mb
Size of Server	3 Mb
60 conn x 50Kb	3 Mb
Min. Proc Cache	1 Mb
Min. Data Cache	4 Mb
Min. Server Memory	11 Mb

If we assume each connection uses 2 stored procedures: 60*2 plus some = 150

If we assume proc size is 5Kb: 150 procs * 5Kb = 750Kb, round *up* to 1 Mb. Procedure cache represents 20% of total cache. Thus data cache is 4Mb.

configuration notes

The above calculations do not take into account other memory-consuming parameters. However, the significant factor (user connections) is included, and shows that at least 11Mb would be required for this server to run as defined. Given the amount of memory available on the system, it would make sense to add significantly more.

configuration commands

To configure these minimum values:

```
exec sp_configure memory, 5632
exec sp_configure "user connections", 60
```

actual values

Values calculated above are estimates only. Actual values in error log.

Other Memory-Related Options

- Maximum number of open objects & open databases
  - Each object uses about 300 bytes of memory
  - Each database uses 7K of memory
  - Set open databases to the number of databases on SQL Server
  - Examples:

```
sp_configure "open objects", 500
sp_configure "open databases", 50
```

- Locks
  - Number of available locks
  - Typical transactions may have 20 locks
- Setting these high does not have a significant impact on performance
  - Increase their value if you get related error messages

Lab 6a: Configuration

In this lab, we ask you to display configuration options and interpret the output.

We also have several thought exercises, in which we ask you to:
(a) predict the effect changing user connections would have on procedure and data caches,
(b) determine what configurations may need adjusting based on error conditions,
(c) determine minimal and maximal configurations given a set of requirements.

1. Use a stored procedure to display configuration values.
2. Are there any zeros (0) in the config_value column? What does this mean? default
3. Which values are currently in effect? config_value Which will take effect after the next reconfigure/restart? run_value
4. Given the current configuration, what effect would there be on the procedure and data caches if you increased the number of user connections by 8? (Predict the number of pages you will be losing in each.) procedure cache = 50%
   8 * 50K = 400K = 200 pages } 100 pages ea
5. You encounter the following problems. Assuming the problem to be one of configuration, how would you fix it?
   a) Lots of physical reads (via set statistics io on)
   memory ↑ procedure cache ↓
   b) Page fault count goes up
   memory ↓
   c) User can't connect
   user connections ↑
   d) "Msg 701: There is not enough procedure cache to run this procedure, trigger, or SQL batch. Retry later, or ask your SA to reconfigure SQL Server with more procedure cache."
   procedure cache ↑

(continue to next page)

Lab 6a: Configuration (cont.)

6. Assume the following:

   Total Mem = 64 Mb, OS needs 2 Mb, Misc. programs need 2 Mb
   Three database devices to be mirrored, eight devices total
   There are 25 SQL Server logins, with only 10 logged in to SQL Server at any one time
   Of these, expect 2 application developers concurrently, each using a single connection
   Expect 8 application users
   Application uses 3 connections
   Application uses two stored procedures (10K, 30K) heavily.
   Expect all stored procedures to be used concurrently by all application users

   Based on these specifications, what are the minimal and maximal configurations for the following options:

   memory
   user connections (50K per user connection)
   devices (512 bytes per device)
   procedure cache

   How much does each set of configurations leave for data cache?

   (See next page for a summary of space usage for various configurable options.)

Configuration Summary for the Lab

Kernel & Server structures:

user connections	50 Kb per
devices	512 bytes per
open databases	35 bytes per
open objects	40 bytes per
locks	72 bytes per

Reconfiguring Dynamic Options

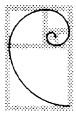
- Some configuration options can be reconfigured "dynamically", without restarting SQL Server
- Partial list:
  - recovery interval, remote access, default language, password expiration interval, nested triggers
- Syntax:

```
sp_configure "option", new_value
```

- Sample:

```
sp_configure "remote access", 1
```

- Then SA must use `reconfigure` to install

SYBASE 

Recovery Interval (Dynamic)

- Maximum number of minutes per database that SQL Server should use to recover, should that be necessary
  - Set in minutes; default is 5
- SQL Server uses this value to evaluate when to write changes to disk
  - Looks at the number of rows added to the log since the last write
- Example:

```
sp_configure "recovery interval", 3
go
reconfigure
go
```

- When might you raise or lower the value of recovery interval?
- What are the trade-offs?

when might you raise or lower recovery interval?

- lower recovery interval if you're expecting an increase in update activity

trade-offs

- lower recovery interval leads to more frequent writes, shorter recovery time likely in case of failure
- higher recovery interval leads to less frequent writes, longer recovery time likely

Other Tunable Options (Dynamic)

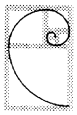
- Remote Access
  - Controls access from remote SQL Servers
  - If set to 1, access is allowed
- Default Language
  - Language used to display system messages (unless overridden by user)
  - us_english id is 0

remote access

- See module on Other Tasks for more detail on controlling remote access.

default language

- Languages must be installed.
- Uses *langid* (language id) which is assigned when language is installed.

SYBASE 

Other Tunable Options (Dynamic)

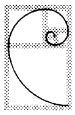
- Password Expiration Interval
  - Number of days passwords are valid; default is 0 (no expiration)
  - Set by SSO
- Nested Triggers
  - Determines whether triggers can call other triggers
  - Default = 1 (nested triggers allowed)
  - Most common configuration: nested triggers allowed

password expiration interval

- Users are warned before expiration.
- If password has expired, user can log in and execute `sp_password` *only.*
- Setting password expiration interval changes a status bit in *syslogins.*

Other Tunable Options (Not Dynamic)

- Database Size
  - Used to determine default database size
  - Can be overridden in `create database` command
- Recovery Flags
  - Affects output to errorlog and console during recovery
  - 0 = minimal output (default); 1 = "noisy" output
- Fillfactor
  - Determines how full index and leaf are at index creation time
  - Affects lock contention & page splitting
  - Can be overridden in `create index` command
  - Rarely necessary to set Server-wide using `sp_configure`

SYBASE 

Other Tunable Options (Not Dynamic)

- Max Online Engines
  - Controls number of SQL Server engines in symmetric multiprocessor environments
  - Default is 1
- Default Network Packet Size
  - Along with "maximum network packet size" and "additional netmem", allows SA to set size of network packets
  - Useful for large data transfers: bcp, read/writetext, large selects
- Audit Queue Size
  - Controls how many audit records are held in memory
  - Trade-off: performance vs. risk
  - Set by SSOs

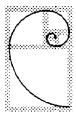
max online engines See discussion of Symmetric Multiprocessor environments in the module on Other Server Administration Topics.

default network packet size See discussion of Bulk Copy in the module on Other Server Administration Toipcs.

audit queue size See module on Auditing.

What to do if...

1. You add three more databases....
2. Users complain that they cannot log into the server...
3. Recovery takes a long time...
4. You reconfigured memory, and now SQL Server won't start up!

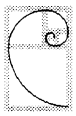
SYBASE 

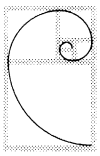
Summary

- Configuration is critical to good performance and good use of resources
  - Most important: memory
- Memory usage depends on
  - user connections, devices, etc.
  - procedure cache
- `sp_configure` and `reconfigure`
  - System-wide; affects all databases
  - Several options are dynamic; all others require rebooting

Partial List of Configuration Options: Units & Defaults

Option	Unit	Default
memory	# of 2K pages	platform-specific
user connections	# of connections	platform-specific
devices	# of devices	10
procedure cache	% of memory for procedure cache	20
open objects	# of objects	300
open databases	# of databases	10
locks	# of available locks	5000
recovery interval	# of minutes per database	5
remote access	0 or 1	0 (no remote access)
default language	language id	id of language chosen at installation
password expiration interval	# of days	0 (no expiration)
nested triggers	0 or 1	0 (no nested triggers)
database size	# of Mb	2
recovery flags	0 or 1	0 (minimal output)
fillfactor	0 through 99	0
max online engines	# of engines (SMP)	1
default network packet size	bytes	512
audit queue size	# of audit records	100





SYBASE®

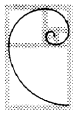
Module 7

Transaction Management

Objectives

- Explain how changes are recorded in the database
- Explain what happens during automatic recovery

SYBASE



Two Kinds of Failures

- System failures, power outages, SQL Server failures
  - SQL Server recovers automatically, using the transaction log
- Media failures
  To protect your database against a media failure:
  - mirror devices
  - use a warm or hot standby
  - make physical backups of your data and logs
- This module focuses on how automatic recovery works

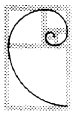
hot standby Up-to-the-minute copy of data. Example: mirrors.

warm standby A warm standby is a copy of the data that is relatively up-to-date. Example: load dumps from one SQL Server onto another. The data on the second SQL Server is relatively up-to-date but not absolutely current.

What Is Recovery?

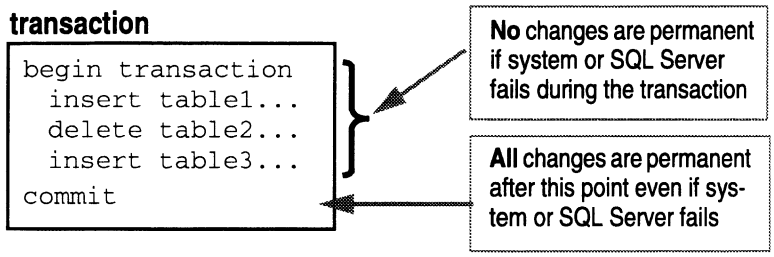
- Automatic effort by SQL Server to ensure database integrity after a system or SQL Server failure, and whenever SQL Server is restarted
- Depends on the following concepts:
  - Transactions
  - Transaction logging
  - Checkpoint
  - Write-ahead log

SYBASE



What Is A Transaction?

- A logical unit of work comprised of a SQL operation or a sequence of SQL statements against the database



- Other users see these changes to the database as a unit
  - They see the data either before any changes or after *all* changes

Transaction

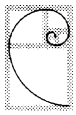
The SQL operation(s) in a transaction:

- Usually includes at least one data modification statement
- Transforms the database from one consistent state to another

How Are Transactions Logged?

- All changes to the database are logged in a table called *syslogs* in each database

  - Physical log (not logical), containing offset and changed byte, not entire page
  - Typically whole rows
- *syslogs* contains enough information to roll back (undo) or roll forward changes if necessary
  - It does not record queries, as these are read-only

SYBASE 

What Are Checkpoints?

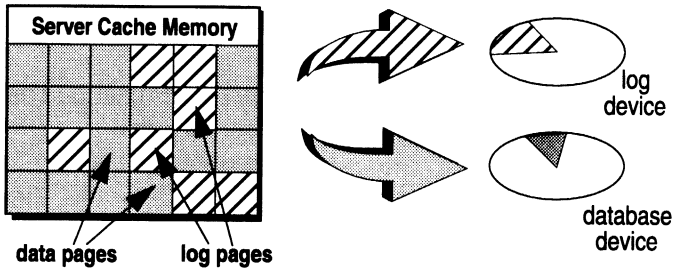
- Checkpoints write log pages and changed data pages from cache to disk
  - They shorten work at recovery time
- What a checkpoint does:
  - Stops new transactions from starting in the database for an instant, typically less than a second
  - Writes "dirty" log and data pages to disk
  - Writes a checkpoint record in the log
  - Allows current transactions to continue without interruption
- Checkpoints are done automatically by SQL Server based on recovery interval
  - dbo or SAs can execute `checkpoint` as well

"dirty" pages

- pages that have been updated since they were last written

What is the Write-Ahead Log?

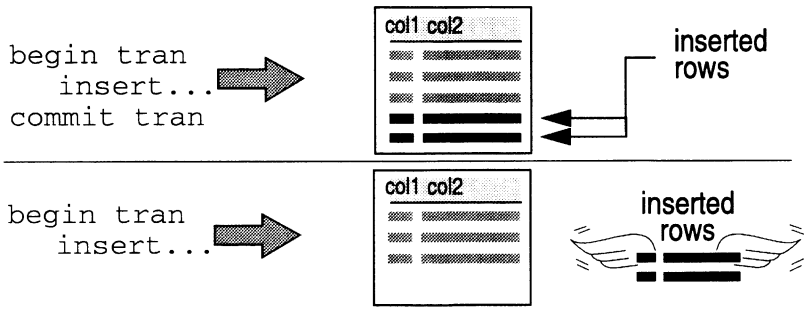
- Log pages are *always* written before data pages



- Log pages written out on commit tran, when space needed, on checkpoint
- Data pages written out when space needed and on checkpoint

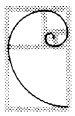
What Happens During Recovery?

- Recovery is performed automatically on each database whenever SQL Server is restarted
  - Guarantees that if a commit took place, those changes will be reflected in the database
  - If a transaction did not commit, the data is changed back to the state it was in before the transaction began



What Recovery Does

- Reads *syslogs* for each database
- Rolls back incomplete transactions (changes data back to state before transaction)
- Checks that committed transactions are present; if not, rolls forward (makes the necessary changes to the data)
- Makes a checkpoint entry in the log
- Logs recovery information in *errorlog* file



Rolling Forward, Rolling Back

When SQL Server is restarted:

If syslogs has...	and database...	then SQL Server...
begin transaction **insert1** **insert2** **(but no commit)**	does not have these rows yet	does nothing
	has some or all of these rows	removes them (rolls back)
begin transaction **insert1** **insert2** **commit**	has some or none of these rows	inserts those not already there (rolls forward)
	has all of these rows	does nothing

- Roll forward = apply changes to database as indicated by the log
- Roll back = remove uncommitted changes from database

Sample contents of errorlog:

```
16:05:38.00 server  Recovering database 'master'
16:05:38.00 server   Recovery dbid 1 ckpt (3047,10) oldest tran=(3047,9)
16:05:38.00 server  1 transactions rolled forward.
16:05:39.00 server   server name is 'FTTOR1'
16:05:39.00 server  Recovering database 'model'.
16:05:39.00 server   Recovery dbid 3 ckpt (383,17)
16:05:39.00 server   Recovery no active transactions before ckpt.
16:05:39.00 server   Clearing temp db
16:05:43.00 server  Recovering database 'pubs2'.
16:05:43.00 server   Recovery dbid 4 ckpt (710,14) oldest tran=(710,13)
16:05:43.00 server  1 transactions rolled back.
16:05:44.00 server  Recovering database 'sybsyntax'.
16:05:44.00 server   Recovery dbid 5 ckpt (383,19) oldest tran=(383,18)
16:05:44.00 server  5 transactions rolled forward.
16:05:44.00 server  Recovery complete.
```

Recovery Options

- Recovery Interval = n (minutes per database)
  - Affects how often checkpoint process writes changed pages to disk
  - Time required to run recovery is estimated based on the number of rows added to log since last checkpoint
  - Dynamic option — takes effect after reconfigure command
- Recovery Flags = 1
  - Prints out each transaction which recovery processes
  - SQL Server must be rebooted for this to take effect
- Examples:

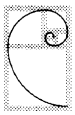
```
sp_configure "recovery interval", 3
sp_configure "recovery flags", 1
```

recovery interval

Approximately once a minute, the checkpoint process checks each database to see how many rows have been added to the log since the last write to disk. It uses this information to estimate how long it would take to recover in case of a failure. If this estimate is longer than the configured recovery interval, the checkpoint process writes the changed pages to disk.

Note that this estimate is usually reliable but can be off by quite a bit, for example when an index is created. The create index command is logged as a single row but could require a lot of time to recover.

In any case, there is no guarantee that recovery interval will be shorter or equal to the configuration value.

SYBASE 

Summary

- Recovery
  - Automatic on SQL Server reboot
  - Ensures that committed transactions are present, uncommitted transactions are not
- Transaction log
  - Stored in each database as `syslogs`
  - Write–ahead
  - Should be kept on separate, mirrored disk

Lab 7a: Transactions & Recovery

In this lab, we ask you to predict the state of your database after a system failure. We provide four scenarios.

1. Assume there is a system failure. The *syslogs* table has the following recorded:

```
begin tran
   delete table1 where...
   insert table2 values (...)
```

   Data on the disk has the `delete` but not the `insert`. What will SQL Server do upon recovery? Which changes will be reflected in the database, which will not?

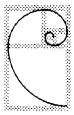
2. For each of the following sets of commands, predict what would happen if SQL Server were shut down and then brought back up. Would the inserted rows be in the appropriate tables?

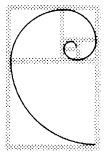
   Student A:
   begin transaction a
   insert tableA values (...)
   commit transaction
   go

   Student B:
   begin transaction b
   insert tableB values (...)
   go

   Student C:
   begin transaction c
   insert tableC values (...)
   checkpoint
   go

   Student D:
   begin transaction d
   insert tableD values (...)
   commit transaction
   go
   checkpoint
   go





SYBASE®

Module 8

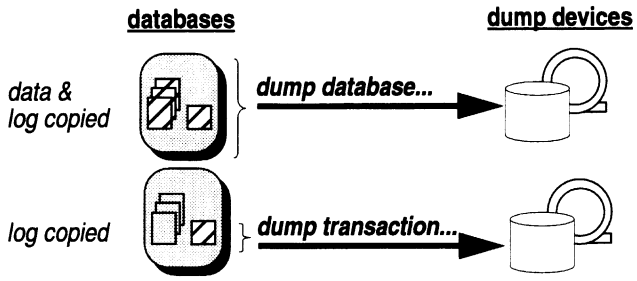
Backing Up Databases & Logs

Objectives

- Back up a database
- Back up just the transaction log
- Use the threshold manager to set up an automatic backup of the transaction log
- Restore a database from backups
- Develop a strategy for regular backups

SYBASE

What is a Backup?



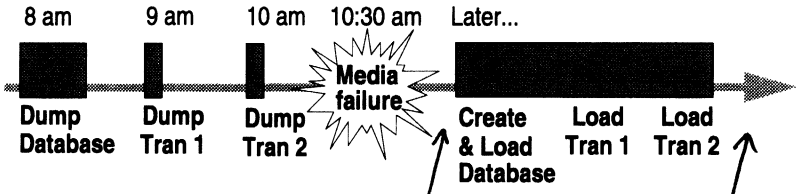
- `dump database` makes a physical backup of entire database, data and transaction log
- `dump transaction` makes a physical backup of the transaction log only
- Do not use OS "copy" commands to make backups!

OS copy commands

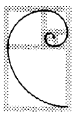
We cannot guarantee recovery if users use OS copy commands. Loading data into SQL Server from an OS copy could lead to massive database corruption.

How to Restore

- If a media failure occurs, you can recreate the databases by loading the backups in the order in which they were created



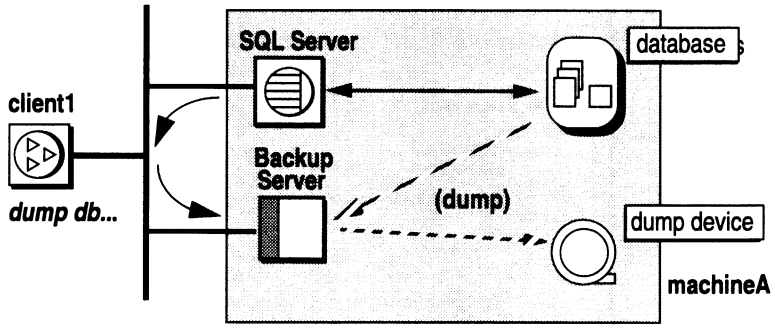
- You can recover up to the state at the time of the media failure (10:30) if the log is on a separate device
  - Otherwise, you can recover up to the state at the last transaction log dump (10 am)

SYBASE 

SQL Server Backups

- All SQL Server backups are performed by an Open Server-based program, Backup Server, running on same machine as SQL Server
- Features:
  - Dynamic--can be done while users are active
  - Can dump to local devices or devices on other machines
  - Can do "multifile" dumps (many databases on a single device)
  - Can do "multivolume" dumps (spread large database across several dump devices)
  - Can set up automatic dumps

The Backup Server



- All SQL Server backups are performed by a Backup Server
- The backup architecture uses a client/server paradigm, with SQL Servers as clients to a Backup Server

how local backups work

Local Backup Server gets instructions from SQL Server (via remote procedure calls) on which pages to dump or load, which backup devices to use, etc., and (Backup Server) performs all the disk I/O. SQL Server does not read or send dump/load data, just instructions.

Unix

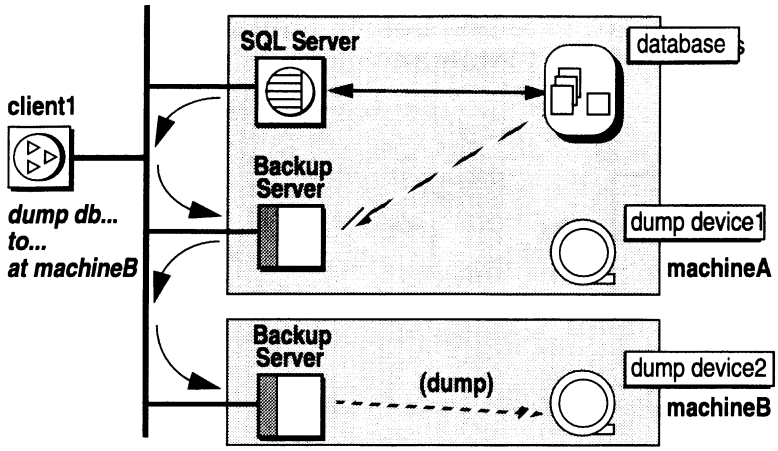
The Backup Server must run on the same machine as the SQL Server requesting the dump or load.

OpenVMS

The Backup Server must be on the same cluster as SQL Server.

SYBASE

The Backup Server: Remote Dumps

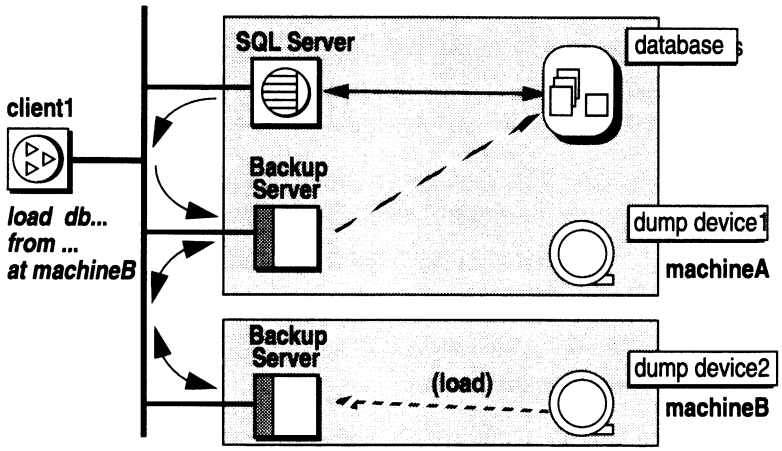


- You can also perform remote backups, in which the local Backup Server communicates with a Backup Server on a remote machine

remote backups: dumps

When you dump to dump devices known by a remote Backup Server, the local Backup Server performs the disk I/O related to the database device and sends the data over the network to the remote Backup Server, which stores it on the dump device.

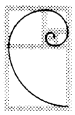
The Backup Server: Remote Loads



- You can also perform remote loads, in which the local Backup Server communicates with a Backup Server on a remote machine

remote backups: loads

When you load from dump devices known by a remote Backup Server, the local Backup Server sends instructions over the network to the remote Backup Server. The remote Backup Server gets the data from the dump device and sends it back to the local Backup Server, which writes them to the database devices.

SYBASE 

Local or Remote Backup Server?

- You always need a local Backup Server
  - The local Backup Server can handle your backup device I/O directly if dump devices are known by the local machine
- If you have many SQL Servers on many machines, you may choose to have them use a remote Backup Server on a network dump host
  - Why? That machine may have a full-time operator or many backup devices
- Each SQL Server has to connect to a local Backup Server in order to use the remote Backup Server
- If local and remote platforms are homogeneous, the volumes written locally and remotely are interchangeable

network dump host A machine dedicated to performing backups for a number of other machines on the network.

Identifying the Local Backup Server

- SQL Server uses the name SYB_BACKUP to refer to the local Backup Server, and by default, it will look for this name in the *interfaces* file whenever it performs a dump or load
- If you would like SQL Server to look for a different Backup Server, specify the name of this Backup Server during installation
- Or, add it to the *interfaces* file and use `sp_addserver` to make it known to SQL Server

```
sp_addserver SYB_BACKUP, null, B_VIOLET
```

name in *interfaces* file

- Be sure to configure SQL Server for remote access

sp_addserver

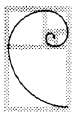
Full syntax:

```
sp_addserver srvname
  [, {local | null}] {,network_name}
```

srvname - always SYB_BACKUP for the local backup server

`local | null` - always `null` for all backup servers (and all remote SQL Servers)

network_name - the name of the local backup server in the *interfaces* file

SYBASE 

Identifying the Remote Backup Server

- To identify a remote Backup Server, add it to the *interfaces* file
  - When you refer to the remote Backup Server name in your dump/load commands, the local Backup Server will look in the *interfaces* file to find that Backup Server
- Remember: *interfaces* entry must be accurate and formatted correctly

Verifying Setup

- Backup Server must be up, else start it running

```
Unix:    startserver -f RUN_B_VIOLET

OpenVMS: startserver /server = RUN_B_VIOLET
```

- SQL Server must be configured for remote access (may need to restart SQL Server)
- The local Backup Server entry in *sysservers* must have an accurate network name

```
sp_helpserver
```

- User who started Backup Server process (usually *sybase*) must have write permission for dump devices

SYBASE

Verifying Setup: *interfaces*

- Both local and remote Backup Servers must be listed correctly in the *interfaces* file

```
#
B_VIOLET
        query tcp sun-ether violet  2001
        master tcp sun-ether violet  2001

#
B_SHANTI
        query tcp sun-ether shanti  9996
        master tcp sun-ether shanti  9996
```

  - Local Backup Server entry created when SQL Server is installed
  - Remote Backup Server (if any): need to add manually
  - If entries are inaccurate, SQL Server may contact wrong Backup Server and write to or read from the wrong devices!

Choosing a Backup Device

- Tapes are preferred because they are more secure
  - Permit a library of database and transaction log dumps to be kept off-line
- Disk files can be used for smaller, less critical databases
  - Faster, but will lose backup if disk fails
  - For this reason, you may want to archive files to tape

SYBASE

Adding a Dump Device (Local Only)

- On most platforms, SQL Server automatically installs in *sysdevices* one or two aliases for tape devices
  - Sample:
```
tapedump1  /dev/nrmt4
tapedump2  /dev/nrst0
```
- To add more, use `sp_addumpdevice`
```
sp_addumpdevice "tape | disk",
  "logicalname", "physicalname", [size
                                  cntrltype]
```
  - Adds a row to *sysdevices*
  - Can be used in subsequent `dump` or `load` commands
- Example:
```
sp_addumpdevice "tape", "tape3",
        "/dev/nrmt4", 3
sp_addumpdevice "disk", "disk1",
        "/usr/backups/disk.dump", 2
```

sp_addumpdevice

Full syntax for tapes:

```
sp_addumpdevice "tape",
device_name, physicalname,
cntrltype [, skip|noskip], size
```

For disks, replace "tape" with "disk" and omit skip, noskip, size

device_name - logical name of the dump device--used in dump/load command

physicalname - physical name of the device

cntrltype - controller number of the device

`skip|noskip` - for tapes, tells SQL Server whether to ignore or recognize existing ANSI tape labels

size - capacity of tape dump device

for local dump devices

You can but do not have to add logical device names for these, as you can specify either physical or logical device names in the `dump/load` commands.

for remote dump devices

You cannot add logical device names for these. The `dump/load` commands expect physical device names for remote devices. (Remote device names are passed to the remote Backup Server by the local Backup Server.)

Lab 8a: Adding Dump Devices

In this lab, you will create two dump devices: dump_devN and logdump_devN. These will be used for database and transaction log dumps in a future lab.

We also ask you to write the complete commands you would use to make certain Backup Servers known to your SQL Server.

The optional exercise asks you to examine files and tables to determine what Backup Servers are known to your SQL Server.

1. Create a disk dump device named *dump_devN*, where N is your user number. The physical device should be a file called *dump_devN*.dat.

2. Create a second dump device named *logdump_devN*. The physical device should be a file called *logdump_devN.dat*.

3. Assume the following situation: You plan to use a local Backup Server called "B_LOCAL" and a remote Backup Server called "B_TUNA" on a machine called "tuna". What commands would you use to make these Backup Servers known to SQL Server? (Do not execute these commands, as we do not have Backup Servers by that name on our system.)

sp_addserver [redacted] syb_backup, null, B_LOCAL

Optional:

[redacted]

4. It's your first day on the job. The former System Administrator has quit without leaving any notes.

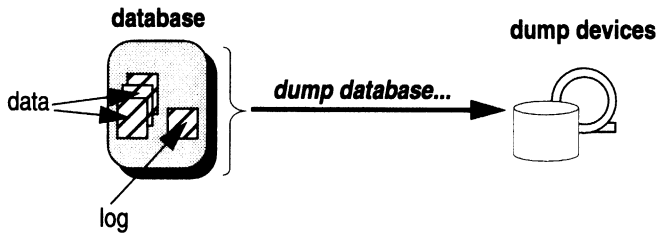
   a) What is the network name of the local Backup Server? SAMAA2_BKUP

   b) Can you tell if SQL Server knows about a remote Backup Server? If so, what is its name, and where is it located?

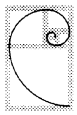
Interfaces file shows 2 entries!

Dump Database

- Makes a backup copy of the database and transaction log



- Dynamic--can be done while users are active
- Sample: `dump database salesdb to data_dev1`

SYBASE 

What Dump Database Does

- Performs a `checkpoint`, meaning log and data are copied from cache to disk (dirty pages only)
- Copies allocated pages (log and data) to dump device
- Captures the state near the *end* of the backup

capturing state near the end of the backup

SQL Server sends the Backup Server a list of all the pages changed by minimally logged operations (such as bcp) that occurred while the backup was in progress, and Backup Server then dumps these.

Then SQL Server instructs Backup Server to dump any log pages that have been written during the backup.

In this way, the backup captures the state near the end of the backup.

Therefore, avoid doing bulk copy, create index or anything that causes lots of page splits during dumps. If you do, SQL Server will have to copy them twice.

Dump Database: Syntax

- Basic syntax:

```
dump database database_name
  to stripe_device [ at b_server_name ]
  [, stripe on stripe_device
     [at b_server_name ] ...
  [with { dumpvolume = volume_name,
     [dismount | nodismount],
     [nounload | unload],  ...
     [noinit | init], }]
```

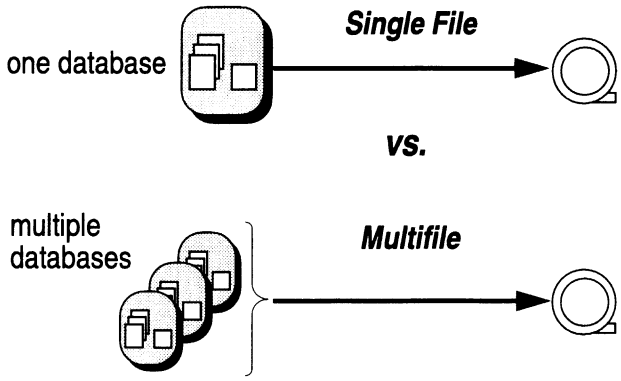
- Major options:

one database per dump device	or	"multifile" dumps "multivolume" dumps
local Backup Server only	or	remote Backup Server
manual backups	or	automatic backups

SYBASE

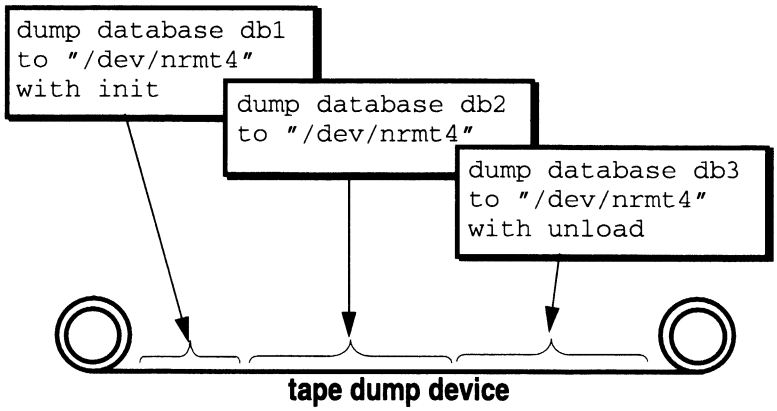
Multifile Dumps--Tape Only

- You can dump more than one database to a single dump tape



Multifile Dumps--Tape Only (cont.)

- Use the `noinit | init` options and `nounload | unload` options to dump several databases to one tape



first

For the first database to be written to a tape, the `dump` command should specify `with init` so that the volume is reinitialized (completely overwritten). `no unload` is the default, which means the tape is *not* rewound and unloading after writing.

second and successive

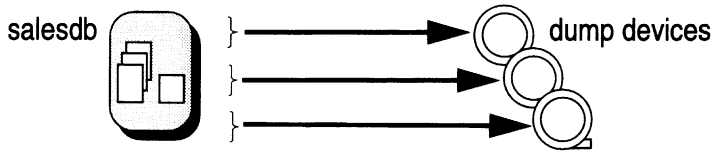
For the second and successive databases (till and including last-1), the `dump` command does not have to specify any `init` and `unload` parameters. The default is `noinit`, meaning the information is appended to the files already on the tape, and `no unload`, so the tape is not rewound after writing.

last

For the last database to be written to a tape, specify `with unload` so that the tape is rewound and unloaded.

SYBASE

Multivolume Dumps (Tape Only)



- You can write a single database across multiple tapes concurrently using the `stripe on` option
- Sample:

```
dump database db1 to "/dev/nrmt4"
  stripe on "/dev/nrmt5"
  stripe on "/dev/nrmt0"
```

  Approximately one-third of the database will go to each tape device

- Operators use `sp_volchanged` (Unix) or REPLY (OpenVMS) to notify Backup Server that the appropriate volume has been mounted

sp_volchanged and disk files

In the case of disk files, you can't change volumes but may still need to execute `sp_volchanged` to confirm, for example, that you want Backup Server to overwrite an existing dump file.

Local vs. Remote Backup Server

- All Backup Server requests are sent to the local Backup Server
- To pass the backup request to a remote Backup Server, specify its name in the `dump database` command
- Example:

```
dump database salesdb
  to "/dev/rmt4" at B_SHANTI,
  stripe on "/dev/rmt5" at B_SHANTI
```

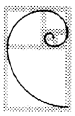
- On most platforms you must use the physical path name for a remote Backup Server; SYBASE logical device names will not work
- Be sure the *interfaces* file entries are accurate for both Backup Servers

Unix

Must use physical path name for remote Backup Server.

OpenVMS

In the case of a remote Backup Server being on another node of the cluster, the system logical will work. If the remote Backup Server is across the network, you must specify the physical path name.

SYBASE 

Manual vs. Automatic Database Dumps

- You can execute dump database commands *manually,* on a certain schedule or when needed
- You can arrange for dumps to occur *automatically* using:
  - OS utilities that can run your dump scripts
  - the threshold manager

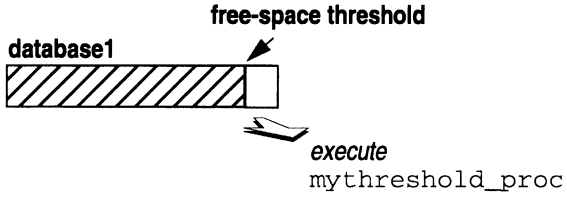
Manual Database Dumps

- Advantages:
  - Useful at smaller sites, or when it's important to know who is performing the dumps
  - Easier to manage tapes
  - Useful if you need to know the file name for loading, e.g., on a multifile volume
- Disadvantages
  - Personnel must be available when backup is to be performed
  - Space usage must be monitored closely, or you may run out of space in database or log

SYBASE

Automatic Database Dumps

- OS utilities can be set up to run dump database scripts at a certain time or at certain intervals
- The threshold manager can be set to execute a dump database command when free space gets low



- Threshold manager: discussed later in this module and in the module on Monitoring & Troubleshooting SQL Server

Dump Database: Samples

```
1)dump database pubs2 to "/dev/nrmt0"

2)dump database pubs2 to "/dev/nrmt0"
     with init

3)dump database pubs2 to "MTA1:",
     stripe on "MTA2:"

4)dump database pubs2 to "MTA0:"
     at B_SHANTI

5)dump database pubs2 to dump_dev
```

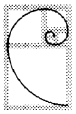
1 Dumps *pubs2* database to dump device. As init is not specified, appends to files already on device.

2 Initializes the tape volume, dumping *pubs2* and completely overwriting existing files on the dump device.

3 Dumps to 2 dump devices, half to each.

4 Dumps to device controlled by named remote backup server.

5 Dumps to the logical device "dump_dev".

SYBASE 

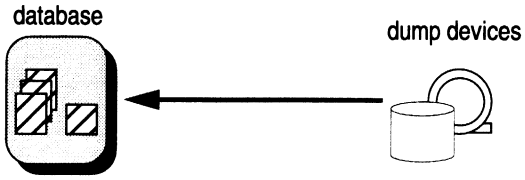
Ensuring Database Consistency

- Although it is not strictly necessary, we recommend ensuring database consistency before dumping the database
  - dbcc checkdb
  - dbcc checkalloc
- It is best to run these in single-user mode; otherwise it may indicate inconsistencies where there are none
- If it is not practical to run these commands (as they are time-consuming), run `dbcc checktable` *`tablename`* and `dbcc tablealloc` for active tables
- More on these commands in the module on Monitoring & Troubleshooting SQL Server

Load Database

- Sample:

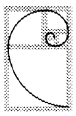
```
load database salesdb from data_dev1
```



- Initializes unused pages in the database and replaces existing contents with the data from the dumped image
  - The owner of the loaded database will be the owner of the database at the time of the dump
- Runs recovery so any uncommitted transactions at the time of the dump are rolled back

ownership

If sp_changedbowner was executed after the backup, it will be invalidated.

SYBASE 

Load Database (cont.)

- Basic syntax:

```
load database database_name
  from stripe_device [ at b_server_name ]
  [, stripe on stripe_device
     [at b_server_name ]
  [with {[dismount | nodismount],
        [nounload | unload] }]
```

- Samples:

```
load database pubs2
  from "data_dev1"
load database pubs2
  from "MTA0:"
load database pubs2
  from "/dev/rmt4" at B_SHANTI,
  stripe on "/dev/rmt5" at B_SHANTI
```

loads from multifile dumps

You do not have to load the first, then the second, then the third dump from a multifile dump--SQL Server will find the named database on the dump device.

loads from multivolume dumps

Multivolume dumps need to be loaded in the right order (i.e., volume 1, then volume 2, then volume 3, etc.).

Load Database Notes

- Database must not currently be in use by anyone
- Can only load dumps made on same machine type
- The database into which you load must exist, and it must be at least as large as the dumped database
  - Use `sp_helpdb` to see amount of space allocated to a database
- Loading data into an existing database overwrites its data; partial loads not possible
- To create a new database for loading, use the `for load` option to `create database`
- If the database is "suspect" (status set in *sysdatabases*), you may have to drop it, then recreate it to load it from a backup

database not in use

You cannot load the database if someone is using it. If you run `sp_dboption "dbo use only", true` and you are the only *dbo,* you can then use *master* and load the database.

create database for load

Ordinarily, `create database` (without `for load` option) builds the system tables and clears unused pages. The `load database` command loads used pages and reinitializes (clears) unused pages. To avoid clearing unused pages twice, use `create database for load`. This command just builds the system tables in preparation for a `load database`.

suspect databases

If a database is marked "suspect", it may be corrupt. In that case, you need to drop it using dbcc dbrepair, recreate it, and then load it.

```
dbcc dbrepair (database_name, dropdb)
create database database_name
  on device = size
  for load
load database database_name
  from dump_device
```

See System Administration Guide, "Recovering Databases", for details.

Lab 8b: Dump Database

In the previous lab, you created two dump devices: dump_devN and logdump_devN. In this lab, you will dump your database to dump_devN. After inserting a new row (i.e., after the dump), we ask you to load the database from dump_devN and examine its contents.

We then ask you to write out the commands needed to perform multifile and multivolume dumps.

The optional exercise asks you to perform a multifile dump.

1. If you do not already have a table called *sm_table*, then create one. Then:
   a. Insert a row.
   b. Dump the database to your dump device dump_devN.
   c. Insert another row, and display the rows of the table.
2. Load the database from the dump device. Use the database, select from the table, and verify that the first row is there, the second is not.
3. Assume the following: the database *salesdb* has been backed up on the tape *public_dev* and is still mounted. There is enough space left on the tape for dumps of *customerdb* and *employeedb*, but no more. What command(s) would you use to dump those two databases onto the same tape as *salesdb* without overwriting the *salesdb* information? What command(s) would you use to load *employeedb*?
4. You have a large database, *hugedb*, that will not fit on a single dump device. You decide to split it across three dump devices: *dump1_dev, dump2_dev, dump3_dev.* What command(s) would you use to dump the database onto those devices?

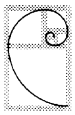
Dump Transaction

- Makes a backup copy of a database's transaction log
- Sample:

```
dump transaction salesdb to saleslog_dev
```

- Incremental backup, used after `dump database`
  - Also prunes the log
- `dump database` dumps but does *not* truncate the transaction log

Note: You cannot dump the log to a device if a minimally-logged operation has occurred (e.g., select into, fast bulk copy) and there has been no intervening dump database operation.

SYBASE 

Dump Transaction (cont.)

- Basic syntax:

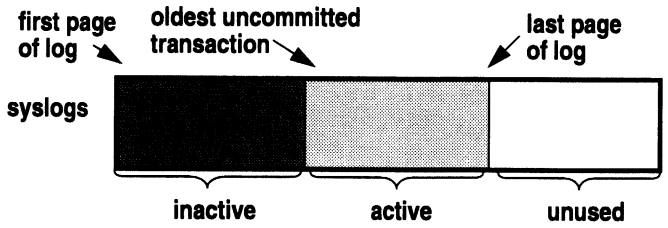
```
dump transaction database_name
  to stripe_device [ at b_server_name ]
  ...
  [, stripe on stripe_device [at
  b_server_name ]
  ...
  [with { dumpvolume = volume_name,
     [dismount | nodismount],
     [nounload | unload],
     ...
     [truncate_only | no_log |
     no_truncate] } ]
```

- Same options as `dump database`, plus "truncation" options

truncation See upcoming discussion.

What Dump Transaction Does

- Dumps entire log and deletes inactive portion of the log (committed transactions)



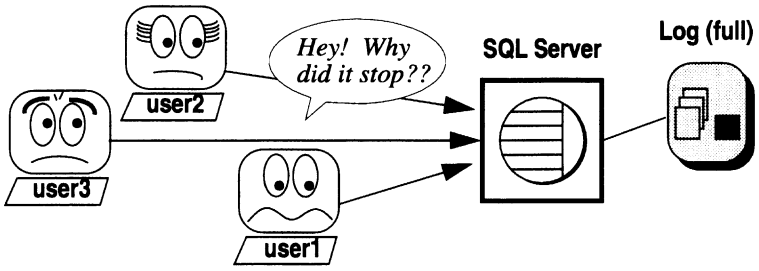
- Truncation options:

with truncate_only	prunes log without dumping
with no_log	prunes log without dumping, and does not log this transaction

SYBASE

What Happens When the Log Fills Up

- The transaction log is appended each time the database is modified — it can fill up quickly
  - If log is full (or the log threshold is crossed and the log is not truncated), all modification work stops



- The log needs to be truncated so this will not happen

threshold

If a log is on its own segment, SQL Server automatically creates a "last-chance" threshold that monitors space usage on that segment. See discussion later in this module.

log full

If the log is not on its own segment and is completely full, you have to execute the following command:

```
dump transaction with no_log
```

Then back up the database immediately!

If SQL Server goes down while `dump transaction with no_log` is running, any action it has taken up to the moment of the falure is unrecoverable, and the database may become corrupted as a result.

How to Truncate the Log

- Several ways to truncate the log
  - `dump transaction` *`database_name`* `to` *`stripe_device...`*
  - `dump transaction ... with no_log`
  - `dump transaction ... with truncate_only`
- If you are not saving transaction log dumps, you can arrange for the log to be truncated each time the checkpoint checking process occurs (approximately once per minute)

  ```
  sp_dboption "trunc. log on chkpt.", true
  ```

  If you set this option, dump the database frequently, since the log is not being saved

dump transaction

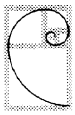
Transaction log is dumped to dump device and log is truncated.

dump transaction with truncate_only

When you use "with truncate_only", the log is truncated but no dump occurs. Use this option when you are not doing transaction log dumps and plan to rely on database dumps.

dump transaction with no_log

When you use this option, the log is truncated but no dump occurs, and there is no record of this in the log. Use this option when you don't have enough space in the log to do a dump transaction with truncate_only.

SYBASE 

Log-Intensive Transactions

- The following transactions could fill up the log:
  - Updating every row in a large table
  - Deleting a large table
  - Inserts based on a subquery
  - Bulk copying in a large table that has indexes
- What could you do in each case to avoid having the log fill up?

updating every row in a large table Break up the query so it affects part of the table at a time, dumping the log after each batch. Then dump the database.

deleting a large table If you do not need a record of the rows, truncate the table instead of deleting the table.

inserts based on a subquery Break up the query, as above, dumping the log after each batch and dumping the database at the end.

bulk copying in a large table Enable truncate log on checkpoint, break up the copy into batches, then (when bulk copy is done) disable truncate log on checkpoint. Be sure to dump the database afterwards.

If the log still fills up in any of these cases, break the query up further, or increase the size of the log.

These transactions are discussed in detail in the Troubleshooting Guide in the section on "Managing Large Transactions."

Monitoring the Transaction Log

- Monitor log usage using the following tools (to be discussed in the module on Monitoring & Troubleshooting SQL Server):
  - `sp_spaceused syslogs`
  - `select data_pgs (8, doampg) from sysindexes where id = 8`
  - `dbcc checktable (syslogs)`
- Use the threshold manager to
  - notify you when the log is filling up
  - do automatic, unattended backups when a certain threshold is crossed
- Check for long-running or "stranded" transactions that keep the log from being truncated (see Monitoring & Troubleshooting)

sp_spaceused syslogs

Reports space reserved, amount used, available space. Not as trustworthy as dbcc, as it may be fooled if the table is inconsistent.

select data_pgs(8, doampg) from sysindexes

Fastest way to determine how full the log is. May be off by as many as 16 pages.

dbcc checktable (syslogs)

Checks pointers, etc., and counts pages.

To get a report on the amount of log space used and what percentage of the log is free, the log must be on its own segment.

This command uses a lot of overhead. Run when the system is not being used heavily.

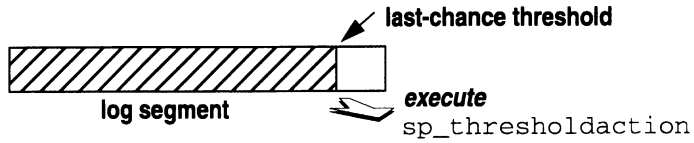
threshold manager

Last-chance thresholds, free space thresholds. Highly recommended. See upcoming pages.

SYBASE

Last-Chance Thresholds

- If a log is on its own segment, SQL Server automatically creates a "last-chance" threshold that monitors space usage on the segment
  - SQL Server places this leaving room for recording log dump



- When the threshold is crossed, SQL Server
  - aborts or suspends user transactions that try to write to the log
  - sends a message to the error log for each transaction that is suspended
  - executes `sp_thresholdaction`

where last-chance threshold is placed

SQL Server places the last-chance threshold leaving enough room in the log to write the records for a transaction log dump. Placement will vary according to the size of the log, and will change if the log is expanded.

You cannot affect where this threshold is set.

putting the log on its own segment (reminder)

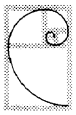
```
create database ... log on device_name
alter database ... log on device_name
sp_logdevice device_name
sp_extendsegment logsegment,
  database_name, device_name
```

Last-Chance Threshold Procedures

- SQL Server does not supply `sp_thresholdaction`—you have to write it!
- Sample threshold procedure that dumps the transaction log and prints a message to the errorlog:

```
create procedure sp_thresholdaction
  @dbname varchar(30),
  @segmentname varchar(30),
  @space_left int,
  @status int
as
dump transaction @dbname to "/dev/rmt4"
print "LOG DUMP:  '%1!' for '%2!'
  dumped",
     @segmentname, @dbname
return
```

@dbname	name of database where threshold was reached
@segmentname	name of segment where threshold was reached
@space_left	threshold size, in 2K pages. corresponds to how much space is free
@status	1 for last-chance thresholds; 0 for all others
`print` statements	Note that `print` statements in threshold procedures will print to the error log. (Print statements in applications go to the user's screen.)

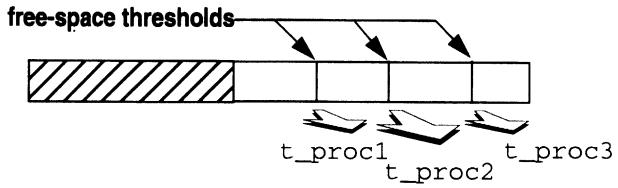
SYBASE 

Last-Chance Threshold Procedures (cont.)

- Some things your last-chance threshold procedure could do:
  - Perform transaction log dumps
  - Write to the error log (using `print` statements)
  - Execute remote procedure calls
  - Expand the log
- Any user with create procedure permission can create a threshold procedure
  - Typically the database owner creates them
- If the last-chance threshold is crossed and SQL Server cannot find *sp_thresholdaction*, an error message is sent to the error log
  - All work in that database is suspended until the log is dumped

Free-Space Thresholds

- You can create additional thresholds, called "free-space" thresholds, on any segment in your database (data or log)



- There can be several free-space thresholds on a given segment
- You need to create a stored procedure associated with each free-space threshold
  - When that threshold is crossed, SQL Server executes that stored procedure

Setting up free space thresholds

See module on Monitoring & Troubleshooting SQL Server for more details

SYBASE

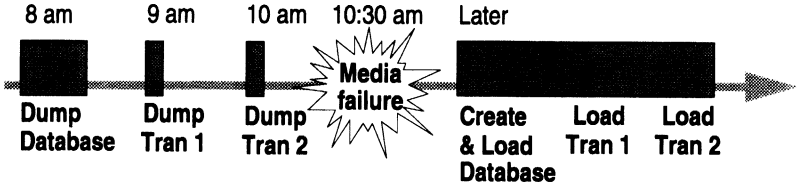
Load Transaction

- Reads the previously backed-up copy of the transaction log into the current transaction log for the purpose of recovering
- Sample:

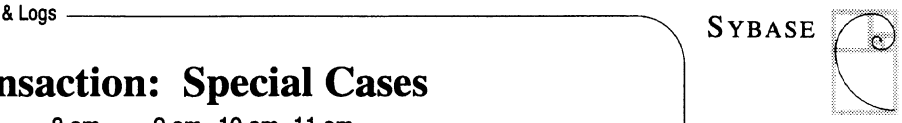
```
load transaction salesdb from logdump_dev
```

DBO-use only!
single-user!
no checkpoint on recovery!

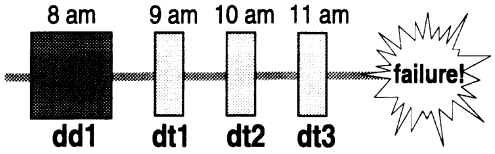
Load Transaction (cont.)



- Transactions must be loaded in correct order, and they all have to be there
- `dump transaction` without a prior `dump database` is useless
- `load transaction` without prior `load database` will fail
- Partial loads are not possible

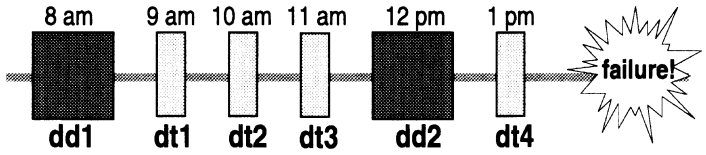


Load Transaction: Special Cases



dd = database dump **dt** = transaction dump

- What if the second transaction log dump (dt2 above) is damaged?



dd = database dump **dt** = transaction dump

- What if dt2 is OK, but 2nd db dump (dd2 above) is damaged?

first scenario

You can recover up to the first log dump, dt1. You cannot "roll through" to the 3rd transaction log dump.

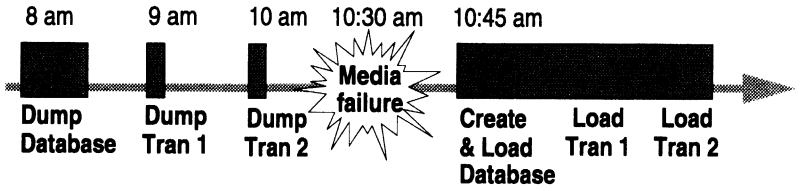
second scenario

You can use dd1, dt1, dt2, dt3, and dt4 to recover the database. In other words, you can "roll through" a missing database dump to restore the database using transaction log dumps.

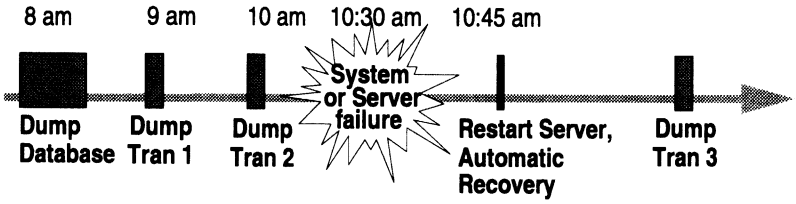
Dump database doet geen truncate op log!

Restoring vs. Recovering

- Restoring from backups...



- Automatic recovery...



Lab 8c: Dump Transaction

In previous labs, you create two dump devices: dump_devN and logdump_devN. You dumped your entire database to dump_devN and loaded the dump back in.

In this lab, we ask you to perform a database dump to dump_devN and a transaction log dump to logdump_devN, then load these dumps back in.

In the last exercise, we ask you to create a last-chance threshold procedure and test it out.

1. In your database, in *sm_table*:
   a. Insert a second row.
   c. Dump the database to your dump device *dump_devN.*
   d. Insert a third row.
   e. Dump the transaction log to your device *logdump_devN.*
2. Load the database from *dump_devN.* Which rows are there? 2 rows
3. Load the transaction dump from *logdump_devN.* Which rows are there now? 3 rows
4. Create a last-chance threshold procedure called *sp_thresholdaction* in your own database, to be executed when your log segment's last-chance threshold is crossed. The procedure should dump the transaction log to logdump_devN and print a message to the error log.
   Test that your procedure works by running an insert/delete batch (see below) terminated by the isql "go 3000", which causes it to execute 3000 times. Check the error log to see if your threshold procedure writes a message.
   batch: insert sm_table values (...)
   delete sm_table
   go 3000
   (We suggest an insert followed by a delete so the data segment doesn't fill up before the log....)

Backup Topics

✓Database dumps and loads

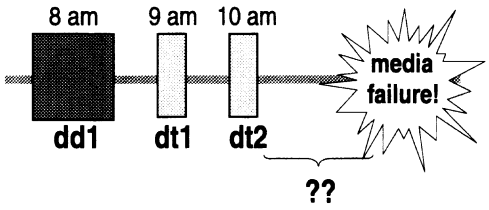
✓Transaction log dumps and loads

- Disaster recovery
  - recovering from the log
  - master database rebuild
- Moving databases to different devices
- Moving databases to different servers

SYBASE

Up-to-the-Minute Recovery

- In the case of a media failure, how can you recover changes that occurred since the last transaction dump?



- If the log and database are on the same physical disk, and that disk fails, you cannot recover these changes
- If they are on separate physical disks, there are 3 possible disk failure scenarios—see next page

Three Scenarios

- If database disk OK, but log disk fails...
  - What if the log device is mirrored?
  - What if the log device is not mirrored?
- If database disk fails, but log disk is still OK...
  - What can you do?
- If both database and log disks fail...
  - What if the log device is mirrored?
  - What if the log device is not mirrored?

database disk OK, log disk fails

If log device is mirrored, SQL Server continues running with the mirrored device.

If the log device is not mirrored, changes that occurred since the last transaction log dump are lost. You can load the database and transaction log dumps up to the latest log dump.

database disk fails, log disk OK

Use the "no_truncate" option of dump transaction to capture the log; then drop the faulty database and rebuild using backup tapes and this log dump.

both database and log disks fail

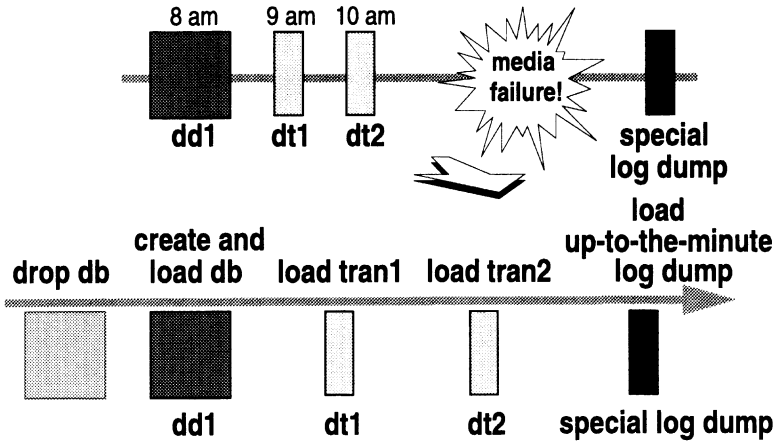
If the log device is mirrored, use the "no_truncate" option of dump transaction; then drop the database and rebuild using backup tapes and this log dump.

If the log device is not mirrored, the changes are lost.

SYBASE

Recovering From the Log

- Using `dump transaction with no_truncate` to get up-to-the-minute recovery is known as "recovering from the log"



recovering from the log

Solution above assumes one of the following:

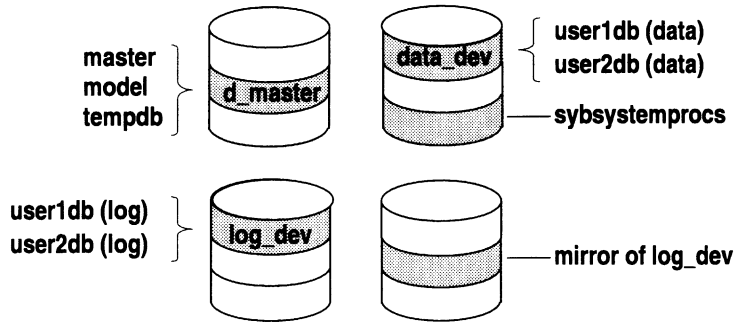
- database disk fails, log disk OK
- both database and log disks fail, but log device is mirrored

Recovering From the Log: Requirements

- Log and database must be on separate physical disks
- The *master* database must be available
- You must execute `dump transaction with no_truncate` *before* dropping the faulty database!

SYBASE

Allocating Devices for Complete Recovery



- Put the log on a separate physical disk, and mirror the log device
- Put master on a separate physical disk

Complete Recovery

To ensure recoverability, all devices should be mirrored including the *master* device.

Backup Topics

✓Database dumps and loads

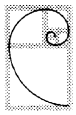
✓Transaction log dumps and loads

✓Disaster recovery

- ✓recovering from the log
- – master database rebuild

• Moving databases to different devices

• Moving databases to different servers

SYBASE 

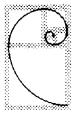
The *master* Database

- The *master* database contains critical information
  - Keep current backups of *master,* especially after the following operations:
    Create/alter/drop database
    Adding or dropping devices
    Adding accounts and users
  - Keep scripts for these as well
- If *master* is damaged or not recoverable, as in the following cases you will need to "rebuild" it
  - If the disk holding master device fails
  - If SQL Server cannot start
  - If dbcc reports damage
- You must also "rebuild" master to duplicate it on another machine

Rebuilding *master* With an Up-to-Date Backup

- If you have up-to-date backups:
  - Use buildmaster –m to build a new *master* database
  - Reboot the server single–user
    Add dump device if necessary
  - Load *master* from backup
  - Restart SQL Server with startserver
  - Change the *sa* password, which is null after buildmaster
  - Check for consistency: run dbcc checkalloc on each database, and examine important tables
  - If all is well, restart SQL Server for multi-user use
- Never execute `buildmaster` while SQL Server is running!

See SYBASE Troubleshooting Guide, Chapter 4: System Database Recovery.

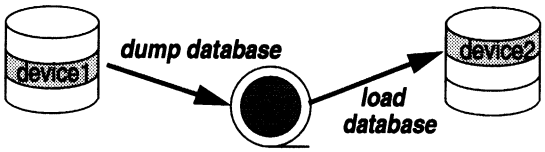
SYBASE 

Rebuilding *master* Without an Up-to-Date Backup

- If you do not have up-to-date backups, then in addition to what has been described in previous pages:
  - Must re–add devices using disk reinit
  - Must reproduce database creates using disk refit
  - Must add all login and user accounts using isql scripts
  - Must reproduce any changes to model and (size of) tempdb
- It's a lot easier to keep up-to-date backups!

See The System Administration Guide chapter on "Recovering Databases" for more detail.

Moving a Database to Another Device

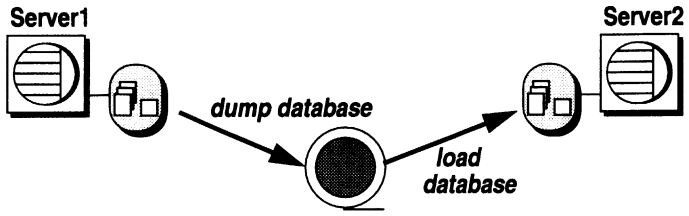


- You can use dump and load to move a database to a different device

```
disk init name = "device2", ...
dump database salesdb to tape_dev
drop database salesdb
create database salesdb on device2
load database salesdb from tape_dev
```

- It is usually not possible to dump and load across platforms
  - Recreate and load the tables manually using bulk copy

Moving a Database to Another Server



- Possible conflicts:
  - suid, uid
  - login name
  - user name
- Be sure the database you create on Server2 matches the original
- Ensure that the target SQL Server uses the same sort order and character set as the original

suid and uid conflicts

To avoid these conflicts, have *dbo* own all database objects, and change the owner of each database to *sa* before dumping the database. Then dump the database and load onto the new Server. Change database ownership back on the new Server.

matching the original database

When you create the database (for load) on the target SQL Server, ensure that you are creating it in exactly the same way, with identical data, log, and segment allocation.

Good Backup Practices

- Develop, document, and test a detailed set of backup and recovery procedures, including the following:
  - Make frequent backups of *master*
  - Dump *master*'s log frequently
  - Keep a current backup of *model*
  - Make frequent database and transaction log dumps for all databases
  - Keep statistics on how long dumps/loads take, and how much space is required
  - Use `dbcc` to verify the integrity of your databases before performing backups

SYBASE

Good Backup Practices (cont.)

- More good backup practices:
  - Dump database after:
    high-speed `bcp`
    `create index`
    `select into`
    `dump transaction with no_log`
    `dump transaction with truncate_only`
  - Automate where appropriate using OS threshold procedures and scripts that run backups
  - Make sure your *interfaces* file is correct
  - Catalogue and label your backup media carefully!

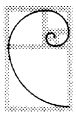
why to dump database

Dump database after above-named transactions because these are only minimally logged. If there were a failure and you did not have an up-to-date backup, you would not be able to recover these changes.

It is not essential to dump the database after `create index`. The index would be recreated if necessary during recovery. However, if you do not dump the database, recovery (if it is necessary) would take longer, as SQL Server would have to recreate the index.

Developing a Good Backup Strategy

- Decisions you'll have to make regarding backups:
  - What backup devices to use, and how to manage them
  - Whether to use a remote Backup Server
  - Whether to have attended and/or unattended backups
- Sample progression, as site needs evolve:
  - Start with manual backups to a single backup device, using a local Backup Server only, on a set schedule, monitoring space usage manually
  - Start striping across multiple devices
  - Automate using OS utilities to run dump scripts, or using the threshold manager
  - Add a network dump host, and install a Backup Server on that machine

SYBASE 

Other Ways to Protect Against Media Failure

- Mirror devices
- Use Replication Server
- Use a warm or hot standby

Summary

- Database backups: dump/load database
  - Performed by Backup Server
  - Multifile, multivolume, local, remote, manual, automatic
  - Uses: media failure, moving databases
- Incremental backups: dump/load transaction
  - Used only in conjunction with dump/load database
  - Various truncation options exist
- Up-to-the-minute recovery possible
- Rebuild master database using `buildmaster`
- Develop, document, test, and time backup and recovery procedures, and manage tapes carefully

Lab 8d: Backup Strategy

In this lab, we ask you to think about what kind of backup strategy you will adopt at your site.

1. Look over the last few pages and decide what backup strategy would be most appropriate at your site.

   Consider:

   (a) what kind of backup schedule you plan to implement for the master and model databases

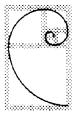
   (b) what kind of backup schedule you plan to implement for user databases

   (c) which backup devices you will use, how you will store and label them

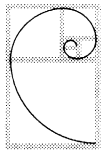
   (d) whether you will make use of a remote backup server

   (e) whether you plan to automate any backups

   (f) whether you plan to mirror devices or make use of hot or warm standby possibilities



Th.A.M. Carters



SYBASE®

System 10

System & Database Administration

Volume 2

Student Guide

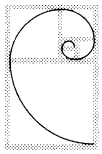
58161-01-0210-01

Notice:

The information contained in this document is subject to change without notice.

Sybase, Inc. MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Sybase, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Sybase, Inc. assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Sybase Inc.



SYBASE®

Module 9

Monitoring & Troubleshooting SQL Server

Objectives

- Monitor SQL Server
- Troubleshoot SQL Server
- List topics in performance and tuning
- Describe protocol for working with Sybase Technical Support

SYBASE

Monitoring Issues & Tools

- system problems
  - error logs, OS monitoring programs
- resource problems
  - error log, sp_helpdb, sp_helpsegment, sp_spaceused, select data_pgs( ), dbcc checktable, threshold manager, sp_monitor
- database integrity
  - dbcc
- performance
  - dbcc, sp_who, sp_lock, sp_monitor
- Tools sold separately:
  - SA Companion, SQL Monitor

System Problems

- i/o errors
- hardware problems
- memory problems
- disk space

Server Resource Problems

- space in log, in database
- memory usage
- number of connections

Database integrity:

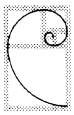
- page linkage problems
- allocation problems
- inconsistent indexes

Performance

- blocking
- load balancing

A Monitoring Strategy

- Monitor system errors, SQL Server error log , Backup Server log
- Monitor space usage using stored procedures, dbcc, and the threshold manager
- Verify/maintain database integrity and validate memory configuration using dbcc
- Monitor overall SQL Server activity using sp_monitor

SYBASE 

Monitoring System Behavior

- Use OS utilities to monitor the SQL Server process
- Example:

  `showserver` (available on most platforms)
- Monitor system error log (if available) to detect disk and memory problems

showserver showserver utility is located in the Sybase install directory.

SQL Server Error Log

- Usually located in the *install* directory in the sybase directory
- Information appended each time SQL Server is started and whenever there is a fatal error or a kernel error
- Error log format:

  date time sender message

- Sample:

```
00:93/06/18 13:40:28.13 server  Recovery
complete.
00:93/06/18 13:50:33.12 kernel  dcreate: error
creating dev100.dat.  File already exists.
00:93/06/18 14:57:22.42 server  Error: 3225,
Severity: 21, State: 1
```

System error log:

Most Unix platforms	/var/adm/messages or /usr/adm/messages
RS6000	SMIT
OpenVMS	analyze /error

Error Messages

- When an error occurs, SQL Server generates an error message:

sample query and error message on user's screen

```
1> select * from titles
2> go
Msg 208, Level 16, State 1:
Server 'VIOLET', Line 1:
Invalid object name 'titles'.
```

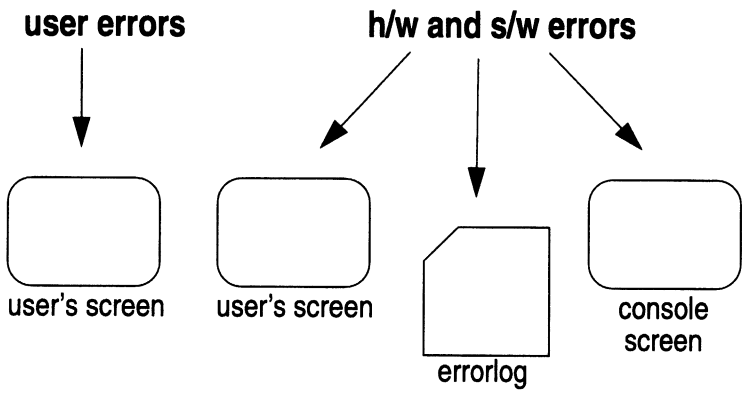
error number, severity level, state

Server name, line number where error occurred

error message

- Supplied system messages are stored in *master..sysmessages*--do not alter these
  - Add user-defined messages using `sp_addmessage`; these are stored in *master..sysusermessages*

Where Error Messages Go

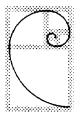


Where do user error messages go?

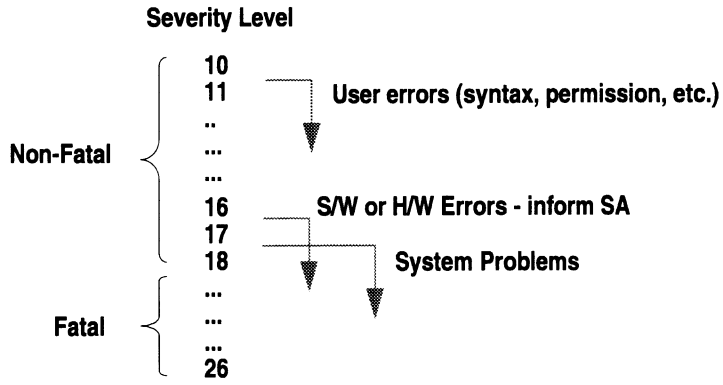
The user's screen (or the application).

Where do hardware and software error messages go?

May go to user's screen, console screen (of the machine where SQL Server was started), or errorlog.

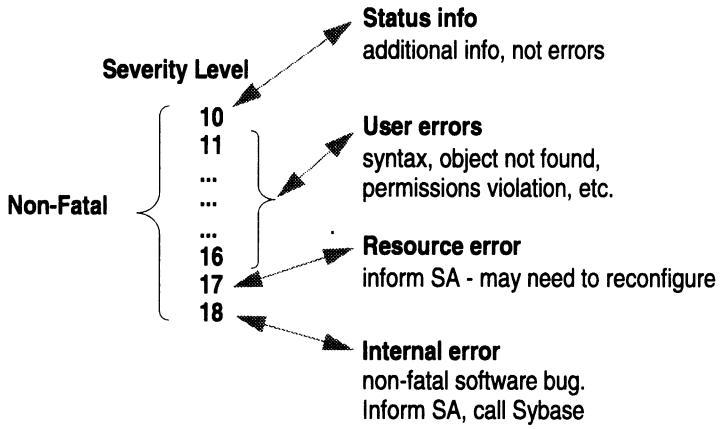
SYBASE 

Severity Levels



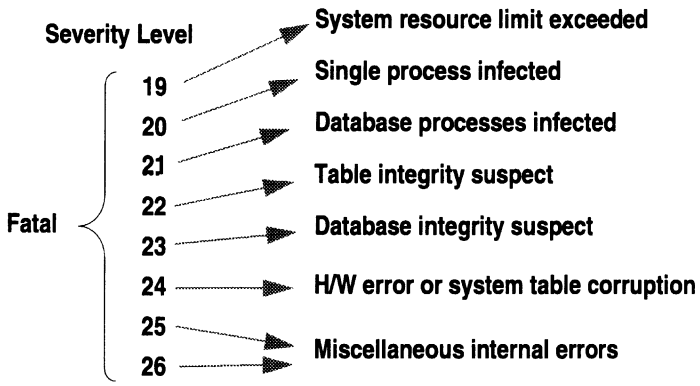
- Monitor error log to look for errors or severity level 17 and above

Recoverable Errors



- Levels 10-16 appear only on user's screen
- Monitor error log for levels 17 & 18 since users may not report these

Fatal Errors

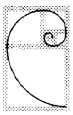


- Where the message does not indicate what to do about the problem, see the Troubleshooting Guide or call Sybase Technical Support

See SYBASE Troubleshooting Guide for a discussion of many errors.

Monitoring the Error Log

- Monitor error log frequently for software and hardware errors (severity 17 and above)
  - Users may not report errors of severity 17 and 18 if their work is not interrupted
  - You can set up a routine that browses the error log searching for specific error numbers or levels
- The error log grows constantly and needs to be pruned regularly
  - Shut down SQL Server first!
  - Make a copy of error log before deleting

SYBASE 

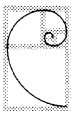
Monitoring the Backup Server Log

- Created when Backup Server is installed
- By default, in *install/backup.log* in sybase directory
- Contains startup information, errors
- Monitor frequently, for example after each full database dump

A Monitoring Strategy

✓ Monitor system errors, SQL Server error log, Backup Server log

- Monitor space usage using stored procedures, dbcc, and the threshold manager
- Verify/maintain database integrity and validate memory configuration using dbcc
- Monitor overall SQL Server activity using sp_monitor

SYBASE 

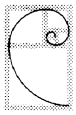
Monitoring Space Usage

- Use the following tools and functions to monitor space usage:
  - sp_helpdb
  - sp_helpsegment
  - sp_spaceused, data_pgs( ), rowcnt( )
  - dbcc checktable
  - threshold manager
- If you run out of log space, selects will work, but modifications won't
  - You will have to dump the transaction log
- If you run out of space for tables and other objects, users who want to insert into existing tables or create objects cannot proceed
  - Free up space by dropping unused objects
  - Alter database to enlarge the database

sp_helpdb

- When used with a parameter (database name), reports information about that database, including free space per device
- Sample output:

```
1> sp_helpdb sales
2> go

name  db_size owner dbid  created      status
sales 4 MB    sa    5     Oct 16 1992 no options
                                       set

device_fragments  size   usage          free kbytes
dev1              2 MB   data only            1376
dev2              1 MB   log only             1008
dev3              1 MB   data only            1008
```

SYBASE 

sp_helpsegment

- When used with a parameter (segment name), reports information about that segment, including the tables and indexes on that segment, and free space
- Sample output:

```
1> sp_helpsegment seg_1
2> go

segment name          status
       4 seg_1        0

device                size              free-space
dev_3                 1MB                      688

table_name          index_name          indid
tableA              tableA                  0
```

sp_spaceused

- Displays free space within tables or in the database
- Sample output:

```
1> sp_spaceused titles
2> go
name   rows  reserved data  index_size unused
----   ----  -------- ----  ---------- ------
titles 18    48 KB    6 KB  4 KB       36 KB
```

```
1> sp_spaceused
2> go
db_name db_size reserved data index_size unused
------- ------  -------- ---- ---------- ------
pubs2    2.0 MB 1386 KB  452  94 KB      840 KB
```

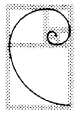
- Expansion space = database size - reserved

reserved Space reserved for objects that have been defined

data Space containing actual data

index_size Space taken up by index(s)

unused Space reserved which does not yet have data



Two Useful Functions

- `sp_spaceused` uses two functions that can be run independently:
  - `data_pgs(object_id, {doampg | ioampg})` estimates pages used by a table
  - `rowcnt(doampg)` estimates rows in a table
- Sample use of data_pgs for *syslogs*:

```
select data_pgs(8, doampg) from sysindexes
  where id = 8
```

- Sample use of `select rowcnt( )` for *titles* table

```
select rowcnt(doampg) from sysindexes
  where id = object_id("titles")
```

- `sp_spaceused` and these two functions will be more accurate if run after `dbcc checktable`

select data_pgs( )

Full syntax of the query given on the foil:
select data_pgs (object_id, {doampg | ioampg}) from sysindexes where id = object_id.

As the object_id of syslogs is always 8, this command displays how much space the log is using.

doampg - page number for the object allocation map of a table or clustered index.

ioampg - page number for the object allocation map of a non-clustered index

In the case of *syslogs*, space usages may be off by as many as 16 pages. This is due to allocations and deallocations in progress while the command is run.

To get the object id of an object, use:

select object_id("objectname")

Example:

select object_id("titles")

dbcc checktable

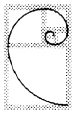
- `dbcc checktable (tablename)` will report most accurately how many data pages are in the specified table
- If the log segment is on its own device, `dbcc checktable (syslogs)` will report the amount of log space used, and what percentage of the log is free
- Sample output:

```
Checking syslogs
The total number of data pages in this table
  is 7.
*** NOTICE:  Space used on the log segment is
  0.01 Mbytes, 1.37%.
*** NOTICE:  Space free on the log segment is
  0.99 Mbytes, 98.63%.
Table has 174 data rows.
```

- Uses a lot of overhead, so do not run while activity is high

listing free space

To list free space by device, see *sp_freedisk* (in "Useful Procedures" in the appendices).

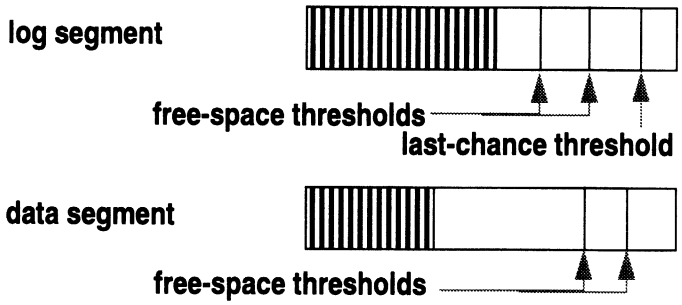
SYBASE 

Monitoring Space Usage

- As we have seen, these tools and functions can be used to monitor space manually
  - sp_helpdb
  - sp_helpsegment
  - sp_spaceused, data_pgs( ), rowcnt( )
  - dbcc checktable
- You can use the threshold manager to monitor space automatically

Threshold Manager

- You can use the threshold manager to notify you when log or data segments are filling up, to dump the transaction log, etc.
  - Use the "last-chance" threshold (log segments only)
  - Set up one or more "free-space" thresholds on data or log segments



last-chance thresholds

If a log is on its own segment, SQL Server automatically sets up a last-chance threshold on it. You can write a threshold procedure (*sp_thresholdaction*) which sends a message to the error log, or dumps the transaction log, etc. as the segment fills up and the threshold is crossed.

free-space thresholds

On segments that contain either data or log, you can set up one or several free-space thresholds, each with its own threshold procedure (named by the user).

SYBASE

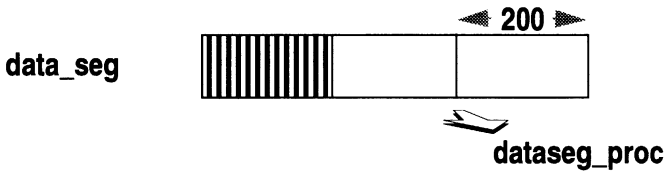
Creating Free-Space Thresholds

- Database owners or anyone with create procedure permission can use `sp_addthreshold` to create a free-space threshold

```
sp_addthreshold database, segment, free_pages,
  procedure
```

- Example:

```
sp_addthreshold salesdb, "data_seg", 200,
  dataseg_proc
```



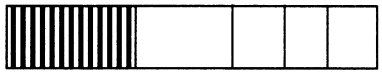
Procedure "dataseg_proc" will be executed when there are only 200 free pages left in "data_seg"

sp_addthreshold This procedure creates a threshold in the named segment *segment* such that the named threshold *procedure* will trigger when the number of free pages in the named database equals *free_pages*.

Creating Free-Space Thresholds (cont.)

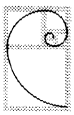
- You can have several free-space thresholds on a given segment

data or log segment



- You can assign separate threshold procedures for each threshold
- Or, System Administrators can write a single threshold procedure, *sybsystemprocs..sp_thresholdaction,* to handle all threshold events
  - The threshold procedure could check input parameters and branch to the appropriate section of code

free-space thresholds You can create up to 256 thresholds per database.

SYBASE 

Displaying Information About Thresholds

- Use `sp_helpthreshold` to view information about thresholds
- Syntax:

  ```
  sp_helpthreshold [segment_name]
  ```

- Without a segment name, displays information about all thresholds in the database
- Example

  ```
  sp_helpthreshold data_seg
  ```

sp_helpthreshold

The output here means that there is a free-space threshold set in *data_seg* that will trigger threshold procedure *data_proc* when there are only 200 pages of free space left.

Status: 0 for free-space thresholds, 1 for last-chance thresholds.

sample output

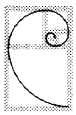
```
segment name     free_pages last chance?
  threshold procedure
--------------- --------- -----------
  --------------------
data_seg              200           0
  data_proc
```

Aborting vs. Suspending User Transactions

- When the log's last-chance threshold is crossed, user processes that try to write to the log are either suspended or aborted
  - By default, they are suspended
- Use `sp_dboption` (in master) to change this:

```
sp_dboption salesdb, "abort tran on log full",
  true
```

  Then execute `checkpoint` in the specified database
- If your threshold procedure dumps the transaction log, would you prefer that user processes be suspended or aborted?

SYBASE 

Thresholds & the Log

- The last-chance threshold:
  - prevents disastrous out-of-space situations, but
  - can still result in suspending or aborting user processes
- Set up a free-space threshold procedure that dumps the log
- Advantages:
  - Minimizes the chances of blocking users
  - Dumps transaction log only as needed
- Arrange for your procedure to write to the error log using `print` statements, and monitor this

writing to the error log

`print` statements in threshold procedures print to the error log, not to the user's screen.

Thresholds & the Log (cont.)

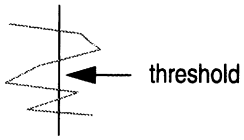
- To place the free-space threshold judiciously, gather data during development:
  - Simulate routine user activity
  - Use the last-chance threshold procedure to record dump times and sizes
  - Monitor the log segment using `sp_helpsegment`
  - Experiment with setting the free-space threshold in different places

System Administration Guide Chapter 10, "Managing Free Space With Thresholds."

SYBASE

When Threshold Procedures Are Triggered

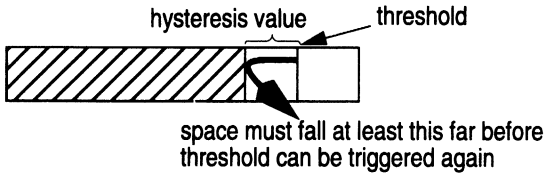
- With frequent inserts and deletes, space in the data segment may fluctuate, and the threshold may be crossed many times



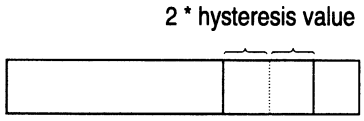
- How does SQL Server keep the threshold procedure from being executed continually?

How Hysteresis Value Controls Thresholds

- To avoid excessive firings, SQL Server uses the "hysteresis value", stored in @@thresh_hysteresis



- When added, a threshold must be at least two times @@thresh_hysteresis pages from the next closest threshold



hysteresis value

This value is set by SQL Server.

example

If one threshold is set at 100 pages and @@thresh_hysteresis is 64, the closest you could place the next threshold would be 100 + (2*64), or 228 pages.

SYBASE

System Procedures for Managing Thresholds

- Use the following system procedures to manage thresholds:
  - sp_addthreshold — Adds a threshold
  - sp_dropthreshold — Drops a threshold
  - sp_helpthreshold — Displays information about thresholds
  - sp_dboption — Sets behavior when log fills
  - sp_helpsegment — Displays information about segment size, space

Lab 9a - Monitoring Space & Using Thresholds

In this lab, we ask you to examine the error log to answer certain questions. Then we ask you to use several tools to determine space usage in your log, and to compare the results.

The optional exercise asks you to set up and test a free-space threshold.

1. Examine the error log and determine the following:
   When was SQL Server last started? 3-9-95 18:46 18:58 06-09 15:50 etc.
   What is its default sort order? bin_roman8 (ID=50)
   Did any errors occur during startup? —
   Have any errors occurred since startup? Yes!

2. Use `sp_spaceused` to determine the space used for the table *sm_table* in your database. 2 KB

3. Use the `data_pgs( )` function to determine the space used for the log in your database. 24

4. Use `sp_helpdb` and `sp_helpsegment` to determine how much free space is left for the log. 976 KB

5. Verify your results using `dbcc checktable`. Comment on any differences.

Optional:

6. Set up a free-space threshold on your log segment that will cause a threshold procedure you write to be executed when the log is half full. (Refer to results above for total log size.) That threshold procedure should dump the log with truncate_only and write a message to the error log indicating the amount of space left in the log. Test that your threshold procedure works by running an insert/delete batch similar to the one you ran in an earlier lab (see below). Check the error log to see if your threshold procedure generates an error message.
   batch: insert sm_table values (...)
   delete sm_table
   go 2000
   (We suggest a dump log with truncate_only to simplify things here. You would normally want to dump to a device!)

A Monitoring Strategy

✓Monitor system errors, SQL Server error log, Backup Server log

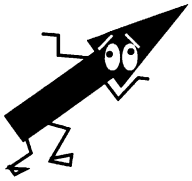
✓Monitor space usage using stored procedures, dbcc, and the threshold manager

- Verify/maintain database integrity and validate memory configuration using dbcc
- Monitor overall SQL Server activity using sp_monitor

SYBASE

Validating Database Integrity

- SQL Server ensures that the internal structures of each database are consistent, but inconsistencies may arise



Msg 2521: Table corrupt: Page is linked but not allocated; check the following pages and table....

Msg 2540: Table corrupt: Page is allocated but not linked; check the following pages and id....

- Use `dbcc` (Database Consistency Checker) on a regular basis to monitor the integrity of databases and objects
  - The sooner inconsistencies are detected, the easier they are to correct

Refer to the System Administration Guide, Ch. 9, "Diagnosing System Problems" for a detailed discussion of how to use `dbcc`.

Checking Tables For Consistency

- `dbcc checkdb` checks all tables in the database for consistency
- `dbcc checktable (table_name)` checks specified table for consistency
- `dbcc checkcatalog` checks system tables for consistency

dbcc checkdb

Sample error-free output:

```
Checking current database
Checking 1
The total number of data pages in this table is 10.
The table has 160 data rows.
Checking 2
The total number of data pages in this table is 5.
Table has 60 data rows.
etc.
```

dbcc checktable(table_name)

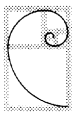
Sample error-free output:

```
Checking tableA
The total number of data pages in this table is 1.
Table has 20 data rows.
```

dbcc checkcatalog

Sample error-free output:

```
Checking current database
The following segments have been defined for database 8
(database name salesdb).
virtual start addr      size      segments
--------------------    ------    ------------------------------
etc.
```

SYBASE 

Checking Page Allocation

`dbcc checkalloc`	• checks page allocation database-wide
`dbcc tablealloc()`	• checks page allocation for specified table
`dbcc indexalloc()`	• checks page allocation for specified table

- All of these commands have an option to fix errors they encounter
- dbcc checkalloc uses a lot of overhead—use during periods of low activity

Using dbcc to Monitor/Maintain Database Integrity

- Time permitting, run `dbcc checkdb`, `dbcc checkalloc`, `dbcc checkcatalog` before each full backup
  - These can be time-consuming--run when activity is low
  - Alternative: load the backup on a different system and run dbcc on the backup
- If you cannot do all this, run `dbcc checktable`, `dbcc tablealloc`, `dbcc indexalloc` on critical tables on a regular basis
- Automate if possible, and read output frequently
- Use `dbcc dbrepair` to drop a database that has been marked "suspect" by recovery

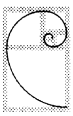
dbcc dbrepair

```
dbcc dbrepair (database_name, dropdb)
```

database marked "suspect"

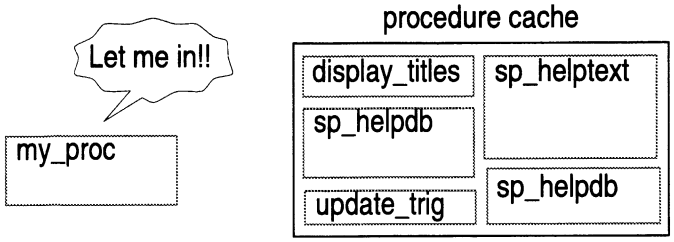
If recovery of a database fails for any reason, SQL Server will mark the database "suspect" to prevent any user from accessing it and potentially making things worse.

Most common reason a database is marked "suspect": the device the database resides on is offline.

SYBASE 

Validating Memory Configuration

- Memory configuration affects performance
- For example, if procedure cache is too small, users will not be able to run stored procedures (or will have to wait)



- Use `dbcc memusage` regularly to check which procedures are being used frequently
  - Use this information to determine best procedure cache size

dbcc memusage

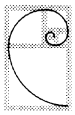
- `dbcc memusage` displays information about how SQL Server memory is being used
  - Turn tracing on first: `dbcc traceon(3604)`
- Output is in three sections:
  - Size of executable code, size of kernel and server structures, size of data and procedure caches
  - Size and number of instances of 20 largest objects currently in data cache
  - Size and number of instances of 10 largest query plans currently in procedure cache
- Monitor over time to determine which procedures are used frequently
  - Use this information to reconfigure procedure cache

dbcc memusage

Tracing: 3604 will send output to the terminal; 3605 will send it to error log.

By getting an idea of which procedures are used frequently, and using information on size and number of instances of each, you can estimate how big procedure cache needs to be and compare that to the actual size. `dbcc memusage` will not tell you, however, whether users have been unable to run procedures.

More on `dbcc memusage` in the Troubleshooting Guide.

SYBASE 

A Monitoring Strategy

✓ Monitor system errors, SQL Server error log, Backup Server log

✓ Monitor space usage using the threshold manager, dbcc checktable, select data_pgs( ), and sp_spaceused

✓ Verify/maintain database integrity and validate memory configuration using dbcc

• Monitor overall SQL Server activity using sp_monitor

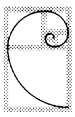
Monitoring Overall SQL Server Activity

- The `sp_monitor` procedure reports information on how busy SQL Server has been
- Most important columns to monitor:
  - cpu_busy
  - total_read
  - total_write
- Establish a baseline and then monitor periodically to look for anomalies

sp_monitor

Sample output:

```
last_run                    current_run                 seconds
-------------------------------------------------------------------
Jul 22 1993 11:12AM    Aug 11, 1993 7:07AM    1713314

cpu_busy              io_busy               idle
-------------------------------------------------------------------
831(825)-0%           2564(2545)-0%         1707710(...)-99%
etc.
```

SYBASE 

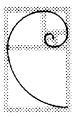
Two Other Monitoring Programs

- SA Companion, useful for system administration functions, has an "auto refresh" capability for displaying processes and locks (similar to `sp_who` and `sp_lock`)
- SQL Monitor can be used to monitor:
  - Data and procedure cache effectiveness
  - Database device activity (for balancing workload)
  - Memory allocation
  - Lock activity
  - Network traffic
  - User Processes (CPU usage, Page I/O, Process state)
- It has both snapshot and continuous data feed capabilities

Note: SA Companion and SQL Monitor are sold separately. Their features will be presented in the module on Other Server Administration Topics.

Developing a Monitoring Strategy

- Data Integrity is most important
  - Adopt good backup habits; run `dbcc` as often as is practical
- Monitoring schedule depends on situation and priorities:
  - database size
  - kinds of queries
  - activity
  - importance of continuous uptime
  - recovery considerations
  - number of users
- Automate where possible, using system files to run the monitoring tools and checking output daily
  - The earlier problems are detected, the easier they are to repair

SYBASE 

Monitoring Rules of Thumb

- Monitor error logs several times a day
- Monitor database integrity before backups, and additionally based on activity
  - With heavy activity, will want to monitor more frequently
  - Identify periods of low activity to run dbcc
- Check the backup log after backups
- Monitor space usage based on activity
- Monitor overall Server activity as required

Lab 9b - Monitoring

In this lab, we ask you to match monitoring tools with what they can monitor. We then ask you to verify database integrity using dbcc.

In the optional exercise, we ask you to consider your company's needs and start planning an appropriate monitoring schedule.

1. Of the following monitoring tools:

system error log	dbcc checkdb
OS monitoring programs	dbcc checktable
SQL Server error log	dbcc checkcatalog
Backup Server error log	dbcc checkalloc
threshold manager	dbcc tablealloc
sp_helpdb	dbcc indexalloc
sp_helpsegment	dbcc memusage
sp_spaceused	select data_pgs( )
select rowcnt( )	sp_monitor

which tool(s) would you use to monitor/detect the following:

whether the system tables are consistent dbcc checkcatalog
how busy SQL Server's CPU has been sp_monitor / OS mon.
space in transaction log sp_spaceused select data_pgs
I/O failures system error log / SQL idea
whether there is a structural integrity problem in the database checkdb
whether there is a structural integrity problem in a specific table " table
whether enough memory has been allocated memusage
hardware problems OS
which stored procedures have been used recently mem
whether pages in the database are correctly allocated checkalloc

2. Check the consistency of your database.

3. Check page allocation in your database.

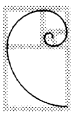
4. Check consistency of system tables in your database.

Optional:

5. Take a moment to think about the specific needs/priorities of your company. How will you monitor SQL Server activity? What tools will you use? On what schedule? Explain your choices.

"What Would You Do If...?"

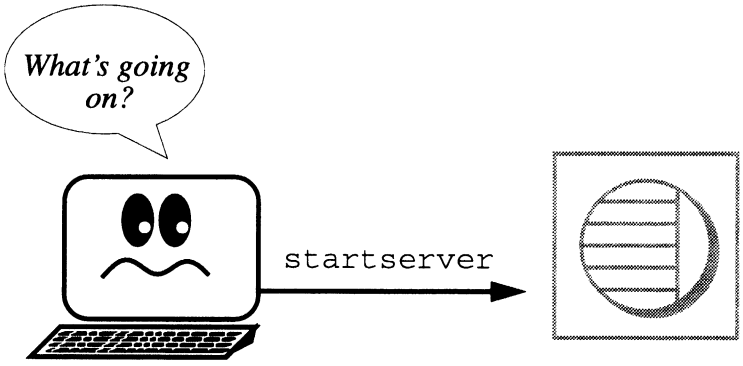
1. SQL Server won't start
2. User "tom", a valid user in the *accounting* database, cannot access that database
3. User "jill" cannot run a query she has run successfully many times
4. Queries that typically take 2 minutes to run are taking 10 minutes to run
5. A user complains that her application suddenly died
6. Only `select` statements work in the *sales* database

SYBASE 

Kinds of Problems

- Server access
  - SQL Server won't start
  - SQL Server is up, but users cannot access it
- Database/object access
  - Users cannot access a database
  - Users cannot access objects
- Performance
  - Processing slows down or hangs
  - Processing stops

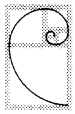
SQL Server Won't Start



- Possible causes? How to investigate?

Possible causes? *(To be filled in by student. Possible answers on the next page)*

How to investigate?

SYBASE 

SQL Server Won't Start: Possible Causes

- SQL Server may already be up!
- Master device may be missing or damaged
- Master database may be inconsistent
- Model database cannot be recovered
- Tempdb cannot be cleared
- SQL Server may be improperly configured
- Operating system may be improperly configured

SQL Server Won't Start: Investigate/Fix

How to investigate:

- Check if SQL Server is already running
  - try to log in
  - run showserver or run OS command to display process
- Check the error log

How would you fix each of the problems on the previous foil?

SQL Server already up? You're OK!

master device damaged? See SYBASE Troubleshooting Guide for directions on how to recover.

master database damaged? Restore from backups. See SYBASE Troubleshooting Guide.

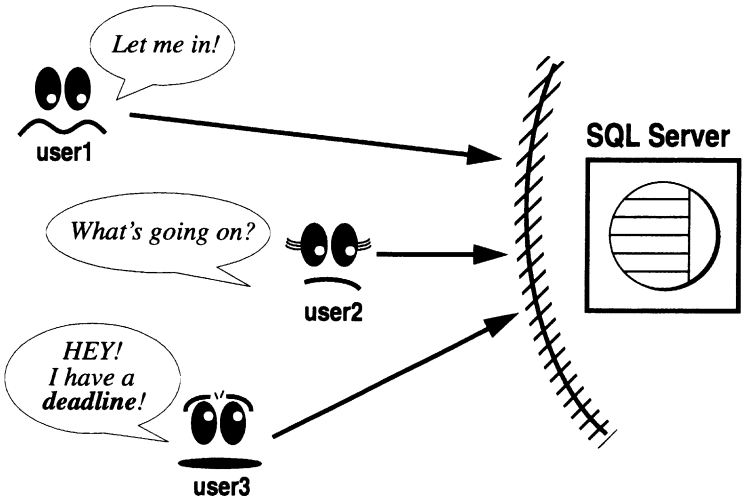
model unrecoverable? Reinstall.

tempdb not cleared? Reinstall, or fix underlying device problem and try again.

poor configuration? buildmaster -r to rewrite config block to default. Then reboot and use sp_configure if necessary.

OS configuration bad? Use Installation Guide and System Administration Guide for guidance on configuring OS.

Server is Up, but Users Cannot Access It

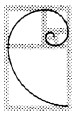


Possible causes?

How to investigate?

Server is Up, But Users Cannot Access It

- Server problems:
  - syslogins inconsistent (rare)
  - SQL Server may be hung
  - SQL Server may be in single-user mode
- Front end problems:
  - network is down
  - no more user connections available
  - interfaces file inaccessible, inaccurate
  - environment variables not set properly
  - client program not working properly
- User error:
  - user using wrong login name or password

SYBASE 

Server is Up, But Users Cannot Access It

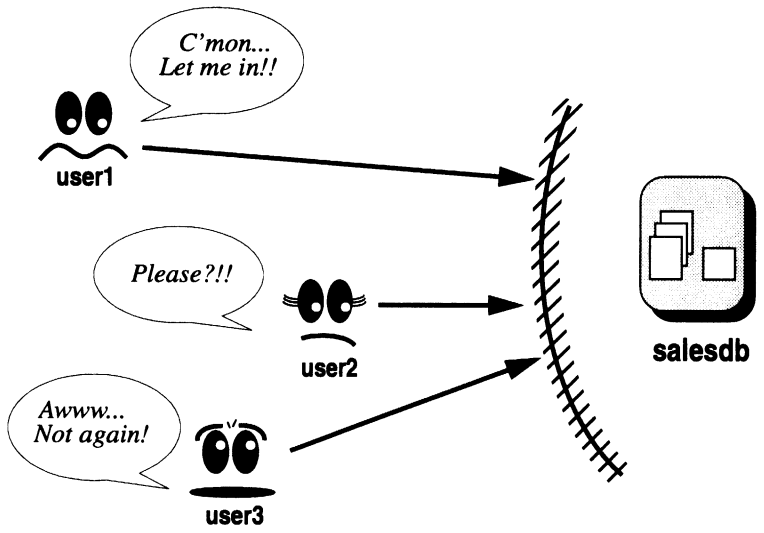
How to investigate:

- Make sure SQL Server is up!
- Check what error message the user is getting:
  - "login incorrect"
  - "SQL Server unavailable or does not exist"
  - etc.
- Try using another client, or accessing SQL Server from another machine
- Are all users having problems, or just some of them? If all users, look in error log and use network monitoring tools.

How would you fix each of the problems on the previous foil?

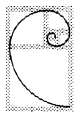
inconsistent syslogins?	Rrecover master from backup. If login is missing, add using `sp_addlogin`.
SQL Server hung?	Should be revealed by OS uilities. Kill at OS level, reboot.
single-user mode?	Reset option or wait until it is reset.
network down?	Investigate? Wait?
no more connections?	Wait, or reconfigure.
interfaces inaccessible, inaccurate?	Investigate, fix.
environment variables not set?	Set properly. See Installation Guide or System Administration Guide Supplement for your platform.
client program not working?	Investigate, fix.
user error?	Investigate, fix.

Users Cannot Access a Database



Possible causes?

How to investigate?

SYBASE 

Users Cannot Access a Database

Possible causes:

- Recovery may have failed, or may still be in progress
- User may not be a valid user in database
- Database options may be in "single user" or "dbo use only" mode
- Database may in the process of loading
- Database may be inconsistent

How to investigate:

- Check error message, error log

How would you fix each of these?

recovery failed? still in progress?

If recovery failed, see error log and consult SYBASE Troubleshooting Guide for more details. If recovery is still in progress, wait.

not a valid user?

Use `sp_adduser` to add login as a user in that database.

"single user" or "dbo use only" mode?

Use `sp_dboption` to reset.

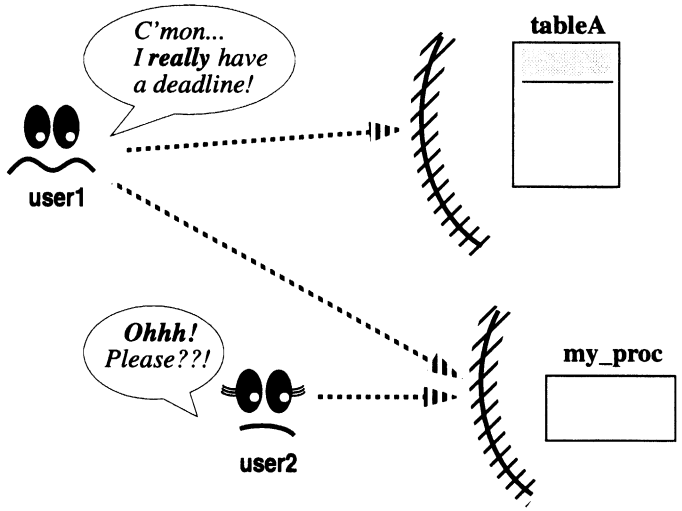
in the process of loading?

Wait.

inconsistent database?

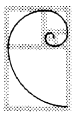
Use `dbcc` to check and fix. May need to recover from backup.

Users Cannot Access Objects



Possible causes?

How to investigate?

SYBASE 

Users Cannot Access Objects

Possible causes:

- Permissions
- Space
- Inconsistency
- Locking

How to investigate:

- Check error message
- For locking problems, run `sp_who` and `sp_lock`

How would you fix each of these?

permissions problems? Use `sp_helprotect` to investigate, and use `grant/revoke` to fix.

space? Did an `update` command fail because of lack of space? Free up space or expand the database.

inconsistencies? Use `dbcc` to check and fix inconsistencies, or recover from backup.

locking problems? See next foil for discussion on `sp_who` and `sp_lock`.

Checking for Blocked Processes

- Use `sp_who` and `sp_lock` to determine what locks are currently being held, if any, and who is blocked by whom

```
> sp_who
> go
spid status  loginame ...blk  dbname   cmd
---- -----    ------   - ---   ------   --
...
6    sleeping  anna      ... 0  salesdb  SELECT
7    sleeping  robby     ... 6  salesdb  UPDATE

> sp_lock
> go
spid locktype      table_id   page   dbname
---- -------       --------    ----   --------
1    Sh_intent     384004399    0    master
6    Sh_table-blk  800003316    0    salesdb
```

- If blocking lasts for any length of time, investigate and/or intervene — possibly kill the offending process

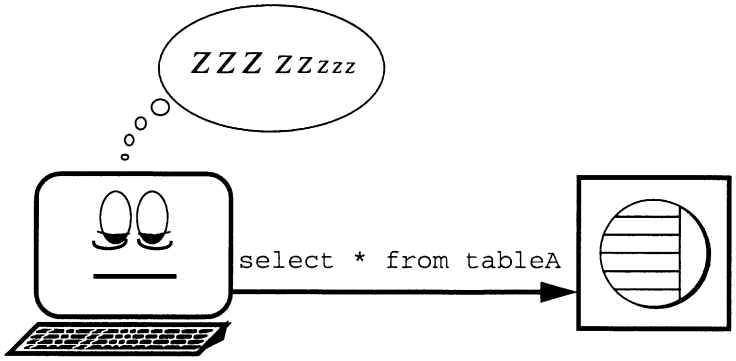
to kill a process

Syntax: `kill` *spid*

Only System Administrators can kill processes, and this permission cannot be transferred.

SYBASE

Processing Slows Down or Hangs



- Possible causes? How to investigate?

Possible causes?

How to investigate?

Processing Slows Down or Hangs

Possible causes:

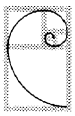
- System problems:
  - using excessive CPU
  - paging
- Locking
- Poor query formation
- Poor database design or choice of indexes for query activity
- Heavy auditing
- Large transactions

transactions that may slow down SQL Server

- create index, large reports, certain monitoring techniques (dbcc, update statistics)

locking

- could be either single, long-running transaction blocking others or multiple processes trying to access the same resource

SYBASE 

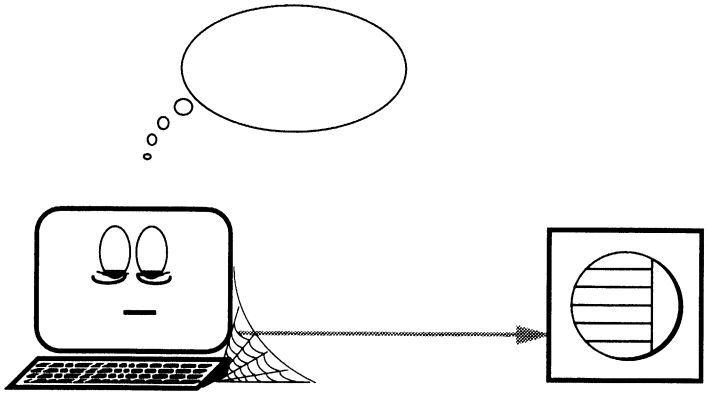
Processing Slows Down or Hangs

- Is system really slow, or is user expecting too much?
- Just one user, or several, or everyone?
  - If one user, what is that user trying to do?
  - If everyone, display SQL Server process
- Run `sp_who, sp_lock`
  - If not blocked, check query or other activity
  - If blocked, check if live process; if live process, contact user blocking resource; kill process if necessary
  - If "stranded" process, try to kill process, or reboot SQL Server; consider activating "keepalive" option if your OS supports it
- Check error log

stranded process

- Also called "ghost" processes, these are processes that are running on SQL Server but are no longer running on the client.
- To determine if a process is stranded, find the spid of the process that is blocking others, then select hostname, hostprocess from sysprocesses where spid = (spid of blocker).
- Then, on that host, display running processes, looking for that one. If it is not there, then it is running on SQL Server but not on the client.
- If the OS "keepalive" option had been enabled, the process may be killed automatically. If not, you can try to kill it using the `kill` command. If that doesn't work, you will have to reboot SQL Server. (Enabling "keepalive" after the fact will not help.)

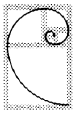
Processing Stops



- Possible causes? How to investigate?

Possible causes?

How to investigate?

SYBASE 

Processing Stops

Possible causes:

- client machine down
- client program down
- disks failed
- server machine down
- SQL Server down
- network down

How to investigate:

- Can you log into Server locally? Remotely?
- Check error log
- If client program went down, check error log for "infected" process

infected process

If the error log refers to an "infected" process:

- determine who was doing what at the time, and
- call Sybase Technical Support for further assistance.

What Else Can Happen?

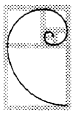
- Error 1105: Can't allocate space for object *X* in database *Y* because the *Z* segment is full
- Symptoms:
  - can't create or modify objects, or
  - can't do any logged operations
- Immediate solutions:
  - if log segment is full, dump it
  - otherwise, expand the database or the log
- Long-term solutions:
  - evaluate log backup schedule
  - analyze space usage: are there objects you can drop?
  - use threshold manager!

dumping the log

Try `dump tran with truncate_only`. If that doesn't work, run `dump tran with no_log`. In either case, back up the database immediately.

expanding the database

Use `alter database` or `sp_extendsegment`.

SYBASE 

If the Log is Full...

- After dumping the log, investigate:
  - Do you need to dump the transaction log more frequently?
  - Has a large transaction (mass update, bulk copy) filled the log?
  - Has a long-running, incomplete transaction filled the log?
- If necessary, expand the log device using `alter database ... log on ...`
- Use the threshold manager to prevent this from occurring

large transaction

- Break these up into smaller batches.
- See Troubleshooting Guide for a discussion of "Managing Large Transactions."

long-running transaction

- Check to see if the log is getting truncated when dumped. The *first* column in *sysindexes* contains the page number of the first page of a given. Check it before and after a dump transaction:

```
select first from sysindexes where id
= object_id("syslogs")
```

  If it does not change, the log was not truncated.
- If no truncation is taking place, there's a long-running transaction getting in the way. Reboot SQL Server or call Sybase Technical Support. (Recovery will remove that transaction from the log.)

expanding the log

To expand the log:

```
alter database salesdb
   log on log_dev = 3
```

Lab 9c - Troubleshooting (Discussion Lab)

In this lab, we ask you to evaluate the problem situations presented at the beginning of this section and come up with possible causes and solutions.

1. For each of the following situations,
   - what the problem might be
   - how you would investigate further
   - how to fix the problem.

   (a) SQL Server won't start.

   (b) User "tom", a valid user in the *accounting* database, cannot access that database.

   (c) User "jill" cannot run a query she has run successfully many times.

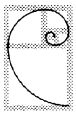
   (d) Queries that typically take 2 minutes to run are taking 10 minutes to run.

   (e) A user complains that her application suddenly died.

   (f) Only `select` statements work in the *salesdb* database.

Performance & Tuning

- System Administrator function
  - Can monitor/tune performance of entire system and/or individual queries
- Performance impact should be considered during development, not just once problems occur
- Performance & Tuning class discusses:
  - Denormalization
  - Indexes
  - Query Optimizer
  - Locking

SYBASE 

Query Tuning

- Where possible, use stored procedures rather than ad-hoc queries
- Efficient SQL
  - Reduce network traffic by batching SQL statements
  - Creating appropriate indexes can improve query performance
- Tools for query tuning
  - `set statistics time on`
  - `set statistics io on`
  - `set showplan on`
  - `update statistics`

Disabling Free-Space Accounting

- For shorter recovery time, you can disable free-space accounting on data segments with `sp_dboption`

```
sp_dboption salesdb, "no free space acctg",
  true
```

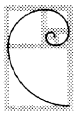
  Then execute `checkpoint` in this database

- Avoid switching this on and off
  - This can lead to inconsistent free-space calculations which will persist until SQL Server is restarted

effects of disabling free-space accounting

If this option is enabled, free-space counts for data segments are not recomputed during recovery. This can be a major time-saver.

Free-space counts for log segments are not affected.

SYBASE 

Reporting Problems to Sybase Technical Support

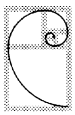
- To improve response time, gather the following information:
  - Site id
  - Environment (platform, SQL Server version, front end, OS)
  - Complete error message and error log
  - What command was running when the error occurred?
  - What other jobs were running?
  - Were there any recent hardware or software changes?
- Determine severity of problem (production down? workaround?)
- Attempt to reproduce the problem
- Only primary or secondary contact should call Technical Support

SQL Server version

To determine what version of SQL Server you are running, execute `select @@version`.

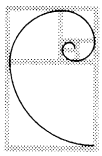
Other Resources

- Insight (dial-in bulletin board service)
- Usenet: comp.databases.sybase
- Compuserve

SYBASE 

Summary

- Good SA techniques include:
  - Regular backups of database and transaction logs
  - Daily monitoring of error logs
  - Pruning of error log and transaction logs
  - Daily monitoring of SQL Server for:
    - space usage
    - integrity
    - performance



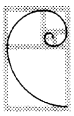
SYBASE®

Module 10

Auditing

Objectives

- Set up and manage an auditing system
- Identify system tables related to system security and auditing
- Define a strategy for Server-wide system security and auditing

SYBASE 

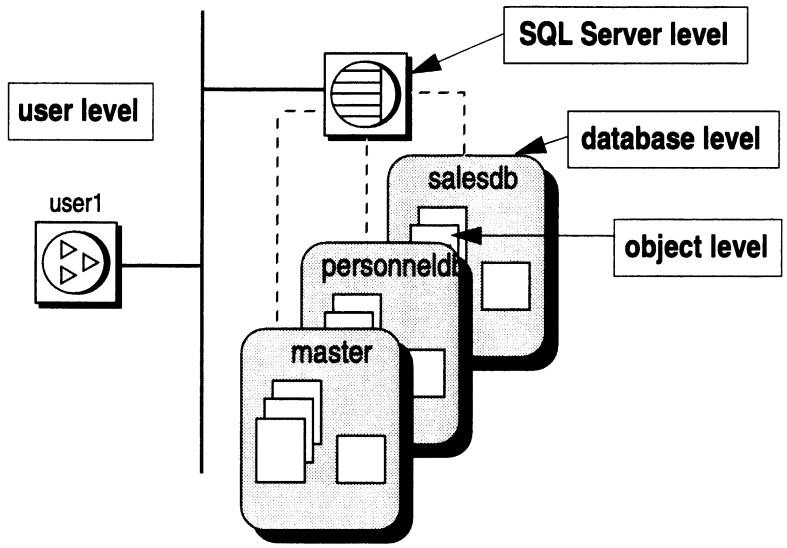
Auditing

- Auditing records security-related system activity
  - Useful for accounting purposes and to track use of authority or resources
  - Traceable to specific logins
  - Conforms to C2 level of security

levels of security

- range from D (lowest) to A2 (most stringent)
- to conform to C2 level of security, a system must have:
  (a) discretionary access control (grant/revoke)
  (b) auditing
  (c) a way to enforce individual accountability

What Can Be Audited



SQL Server level

Some auditable activities:

logins, logouts, reboots, remote procedure calls, fatal errors, privileged commands

Database level

Some auditable activities:

grant, revoke, truncate table, drop, use

Object level

Some auditable activities:

table/view access, procedures, triggers

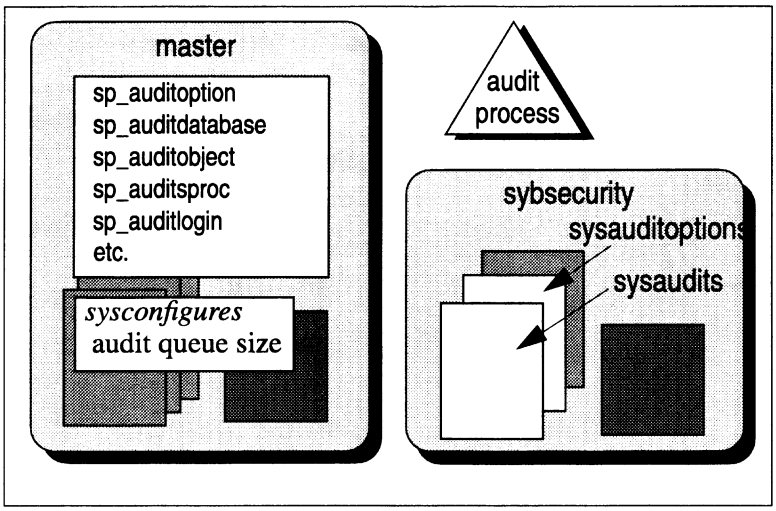
User level

Some auditable activities:

table/view access, command text

SYBASE

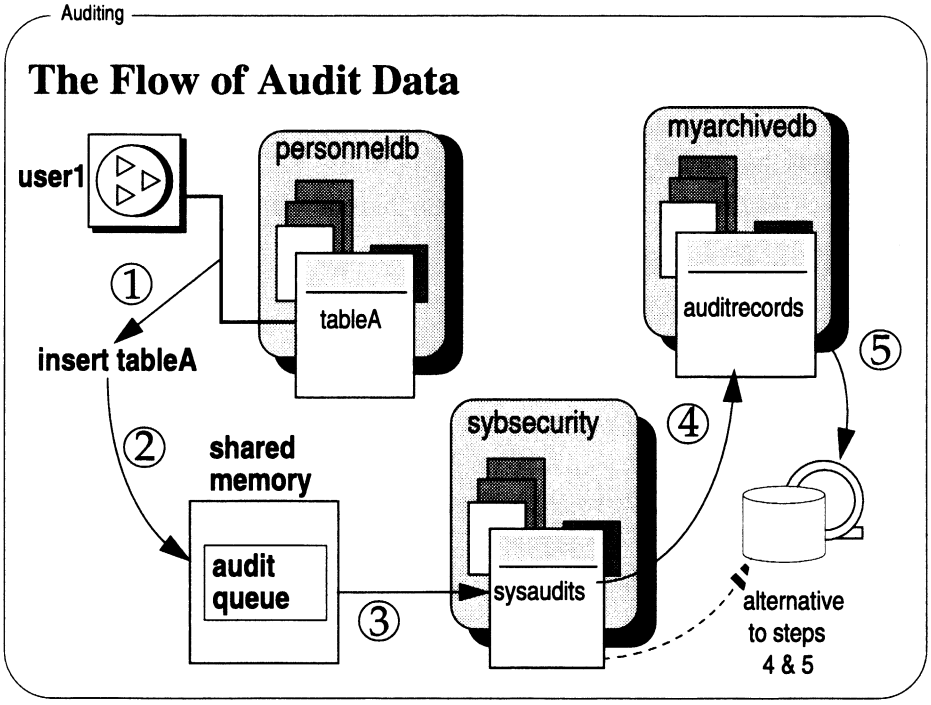
The Audit System



sybsecurity System database that must be installed explicitly for auditing to be possible.

sysaudits Table that contains audit data.

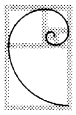
sysauditoptions Table containing option settings related to auditing.



flow of audit data

1. User tries to insert into tableA, and auditing has been set up for this event.
2. After permission checks, information about this command and process is recorded in the audit queue in shared memory (separate from the data and procedure caches).
3. Rows from the audit queue get copied to disk (*sysaudits* in *sybsecurity* database) when the audit task gets its turn.
4. The point of auditing is to save this information in case it is needed. Therefore, at some point you will want to copy *sysaudits* to an archive table (or to tape). This can be done either manually or via a threshold procedure. *sysaudits* must be truncated as well. To copy the data, we recommend using insert into...select, which is logged.
5. Data gets copied to storage media using dump database or bulk copy.

Alternate to 4 & 5: Copy information directly to tape.

SYBASE 

Installing the Audit System

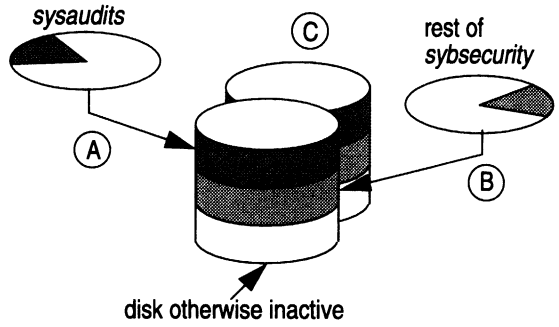
- Install the audit system using `sybinit`
  - You need SA and SSO role to run this command
- What `sybinit` does:
  - Executes `disk init`
  - Creates *sybsecurity* on that device
  - Runs `sp_auditinstall`, creating a segment and placing *sysaudits* on it
  - Runs `installsecurity`, which
    a. creates and populates *sysauditoptions*
    b. creates the audit system procedures
  - Reboots SQL Server
- If you run these commands manually, you need to have the SA role to create a device and the SSO role to create *sybsecurity,* etc.

sybinit

This is the same command used to install SQL Server and Server. sybinit will prompt you for device name and size, and for a segment name.

Choosing a Disk for *sybsecurity*

- The audit trail is write-intensive, so...



- Avoid putting other tables in *sybsecurity*

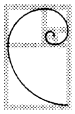
recommendations

A, B: Place *sybsecurity* (and *sysaudits* in particular) on disks that are otherwise not active. Place *sysaudits* on its own segment so you can use the threshold manager to manage it.

C: If audit data is critical, mirror audit devices.

using the threshold manager

If the suggestions outlined above are followed, then the only extensive activity on the device will be writes to *sysaudits*. If *sysaudits* is on its own segment, the threshold manager will work nicely.

SYBASE 

Creating an Audit Trail

1. Choose disk and determine size
2. Install the audit system
3. Configure audit queue size appropriately
4. Arrange for periodic archiving and truncating of *sysaudits*
5. Enable auditing of specific events
6. Enable auditing of Server-wide events
7. Turn on auditing

audit queue size

- configure using `sp_configure`
- default = 100 records
- need to reboot after configuring

auditing of specific events

- database access, object access, attempts to execute stored procedures, etc.

auditing of Server-wide events

- logins, logouts, reboots, RPCs setting roles, etc.

System Procedures For Auditing

sp_auditdatabase	events in database
sp_auditobject	access to tables and views
sp_auditsproc	execution of stored procedures and triggers
sp_auditlogin	activity of specified login

- To display current settings, execute the appropriate procedure without parameters

sp_auditdatabase

```
sp_auditdatabase [dbname [, "ok | fail
    | both | off" [, "d u g r t o"}]]]
```

- d = drop; u = use; g = grant; r = revoke; t = truncate; o = "Outside access"--execution of commands from within another database that reference objects in *dbname*

sp_auditobject

sp_auditobject *objname, dbname* [, "{ok | fail | both | off}" [, "{d i s u}"]]

- d = delete; i = insert; s = select; u = update

sp_auditsproc

sp_auditsproc [*sproc_name* | "all", dbname [, "{ok | fail | both | off}"]]

- above establishes auditing for existing stored procedures and triggers; can also set for future procs and triggers

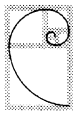
sp_auditlogin

sp_auditlogin [*login_name* [, "table | view" [, "ok | fail | both | off"]]]

- audits *login_name*'s attempts to access tables or views; can be enabled for successful accesses, failed, or both

sp_auditlogin [*login_name* [, "cmdtext" [, "on | off"]]]

- preserves text of all command batches submitted to SQL Server by *login_name*

SYBASE 

System Procedures For Auditing (cont.)

sp_auditoption	global audit options
sp_addauditrecord	add user-defined audit records (comments) to *sysaudits*

- Sample options to sp_auditoption:
  - enable auditing, logins, logouts, server boots, rpc connections, authorizations (setting roles), sa | sso | oper commands, errors
- To display current settings, execute the appropriate procedure without parameters

sp_auditoption

- Note that "enable auditing" must be set for *any* auditing to take place.
- Syntax varies with option being set:

To set all, enable auditing, logouts, server boots, and adhoc records use

```
sp_auditoption "option" [,"{on|off}"]
```

To set logins, rpc connections, and authorizations use

```
sp_auditoption "option"
  [, "{ok|fail|both|off}"]
```

To set auditing for errors use

```
sp_auditoption "errors" [,
  "{nonfatal|fatal|both|off}"]
```

To set auditing for commands requiring special privileges, use

```
sp_auditoption "{sa|sso|oper} commands"
  [,"{ok|fail|both|off}"]
```

sp_addauditrecord

- See Reference Manual for syntax.

Contents of *sysaudits*

- Following are columns in *sysaudits*:

```
event           suid            dbname
mod             dbid            objname
spid            objid           objowner
eventtime       xactid          extrainfo
sequence        loginname
```

- Sample (partial) row from *sysaudits*:

```
event loginname  dbname    objname    extrainfo
-----------------------------------------------
  105 robby      user1db   tableA     SELECT
```

See System Administration Guide Appendices for details on *sysaudits* columns.

Summary

- Your site security policy includes:
  - Whether to have an all-powerful *sa*, or use roles
  - Which users (if any) should have special privileges
  - Which procedures and objects need protection
  - Whether to audit, what needs auditing, and how you will manage the auditing system
- Auditing
  - Server-wide options
  - Database access
  - Object access
  - User activity

auditing plan

- Where do you intend to place *sysaudits*, *sybsecurity*, the "archive" table?
- Will you copy and clear *sysaudits* manually or use the threshold manager?
- What activities do you intend to audit?
- Will you audit the activities of specific users, or activities in specific databases, or activities related to specific objects in the database?
- How will you monitor auditing activity to determine if audit queue size is appropriate, if sysaudits is the right size, if the threshold is well-placed?

Lab 10a - Auditing

In this lab, we ask you how you would install the audit system, and we ask you to determine if an auditing system is installed on your SQL Server. We then ask you to set up auditing of specific events.

1. What command(s) would you use to install the audit system?

2. Check to see if an auditing system has been installed on the SQL Server you are using. If so, what is the value of "audit queue size"? 100 Wel geinst. Niet aktief.

3. Enable auditing of the following events:

   a) All commands submitted by your neighbor (or someone working on your Server).

   b) All attempts to "use" your database.

   c) Attempts to execute procedures in your database.

   d) All attempts to update the *sm_table* table in your database.

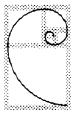
4. Determine if Server-wide auditing is set. If it is not, turn it on and verify that it is set.

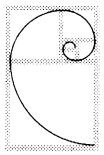
5. Arrange for the audited events to occur, and check *sysaudits* to see if the data is being recorded.

6. Write the full commands you would use to enable auditing of:

   a) the execution of sp_addlogin

   b) all failed login attempts.





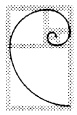
SYBASE®

Module 11

SA Companion

Objectives

- Describe the features of SA Companion
- Examine the forms used in SA Companion to perform SQL Server administration
- Use SA Companion to examine SQL Server, device, database, and user information

SYBASE 

What is SA Companion?

SA Companion is a client tool that

- Provides a forms-based interface for administration of SQL Servers
- Provides interfaces to SQL Server release 10.0 and release 4.9.x
- Enables you to concentrate on what you need to do without having to think about which
  - System tables to examine and
  - System stored procedures to use

SA Companion Functions

The SA Companion modules perform the following functions:

- Server Management: Setup, control, maintenance, and monitoring of SQL Servers
- Device Management: Setup, maintenance, and monitoring of devices
- Database Management: Setup, control, and maintenance of databases, database objects, and database users
- User Management: Setup and control of SQL Server users
- SQL: Creation and execution of SQL statements
- Reports: Generation of standard reports

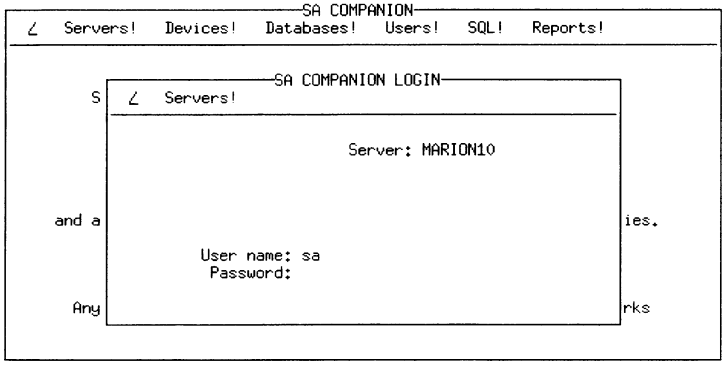
SA Companion

SA Companion functions are grouped into modules, each of which addresses an area where a System Administrator or Database Administrator has responsibility.

SA Companion, like SYBASE's Data Workbench, is an APT application.

SYBASE

Logging in to SA Companion



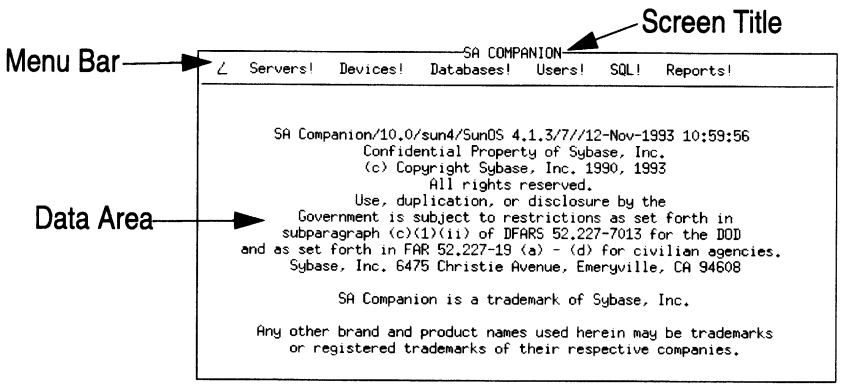
- You can only log into one SQL Server at a time
- You can continue the SA Companion session without connecting to a SQL Server; for example, you may start or stop a Server

Log Into a SQL Server

Multiple servers are supported if the interfaces file contains the names and locations of each server.

SA Companion

Main Menu

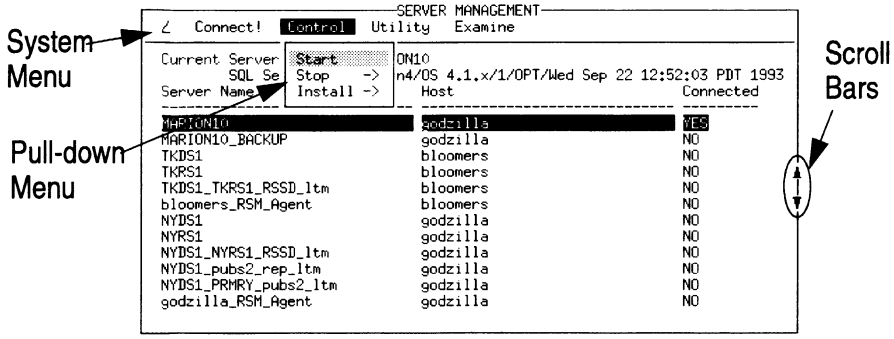


- The Main Menu is comprised of a:
  - Screen Title
  - Menu Bar: Containing the menu selections for the screen
  - Data Area: For viewing and entering data

Main Menu

The Main Menu is the screen from which you access all of SA Companion's functions.

Making a Menu Selection



- You can use a mouse or the keyboard. With the keyboard:
  - Toggle from data area to menu by pressing Ctrl+r
  - Use the arrow keys to move right or left or press the number of the menu
  - Select an item in the pull-down menu by using the arrow keys or type the first letter of the item, then press Return

Menu Selection

- The first menu, the slash (/), is menu 0. "Connect!" is menu 1. "Utility" is menu 3, etc.
- A menu choice with an exclamation point (!) next to it, for example "Connect!", will perform a certain action. Those without an exclamation point display a pull-down menu.
- If a menu or pull-down menu item is not available, it is grayed out.
- If a pull-down menu item displays a "->" next to it, another level of menus, or a cascading menu, is available. For example, selecting the "Install->" pull-down menu item will display a cascading menu.

System (/) Menu

The System Menu is present on every Sa Companion form. Use the System Menu to:

- Access context sensitive help
- Exit a form (move up the menu hierarchy)
- Exit SA Companion
- Print form and report information

Scroll Bars

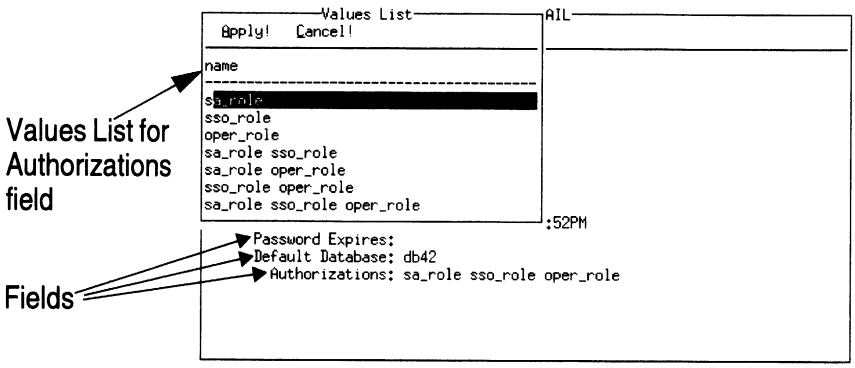
If scroll bar arrows appear, more data exists that is not displayed in the data area. Scroll bar arrows are for vertical scrolling (up and down) as well as horizontal scrolling (left and right).

Using the keyboard:

- To move down or right press Ctrl+f
- To move up or left press Ctrl+b
- To display more information contained in a field, press Return

Sometimes the horizontal scroll arrows appear in the menu bar.

Entering Data



- You either type information in the field or make a selection when there is more than one choice
- Press Tab to move from field to field
- Press Ctrl+v in a field to display a values list, then select an item from the list

Fields

The types of fields are:

- Entry field: Accepts data that you type. You can, also, select data from a values list, if available, for an entry field.
- Choice field: Displays multiple values on the screen. To select a value, move the cursor to the value then press **Return**.
- Rotating choice field: Displays multiple values in sequence. Press Return until the appropriate value appears.

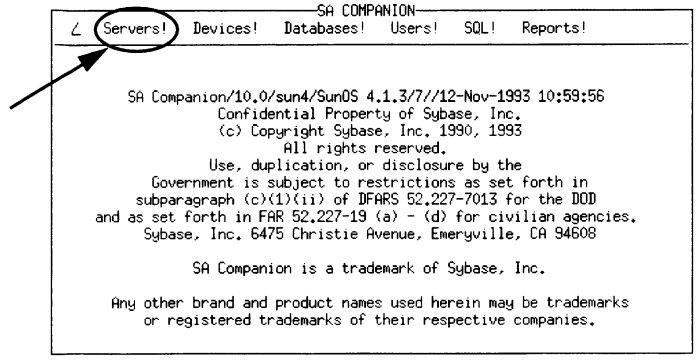
Values List

Values lists displays the valid entries for a field, allows you to select the one you want then transfers the selection to the entry field.

- To make a selection from the Values List, move the cursor to your selection then press **Return**, or select Apply! from the menu bar of the Values List.
- To close the Values List window without selecting a value, select Cancel! from the menu bar.

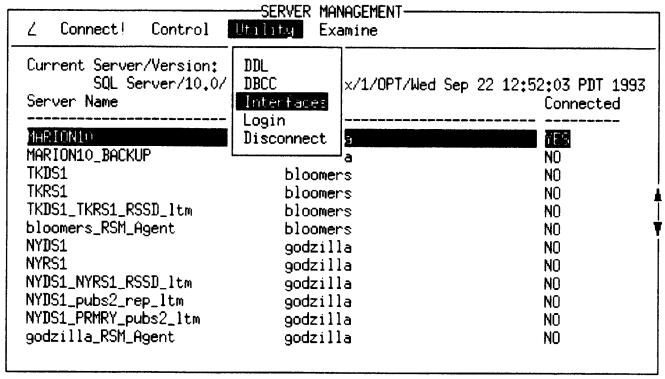
SYBASE

Server-wide Operations in SA Companion



- The next several slides examine the various SQL Server-wide operations that can be performed with SA Companion
- Choose Servers! from the main menu

Connecting to /Disconnecting a SQL Server



- After choosing Server! from the main menu, select the server name in the data area then choose to Login or Disconnect
- The Connected column indicates whether you are connected or disconnected to the Server

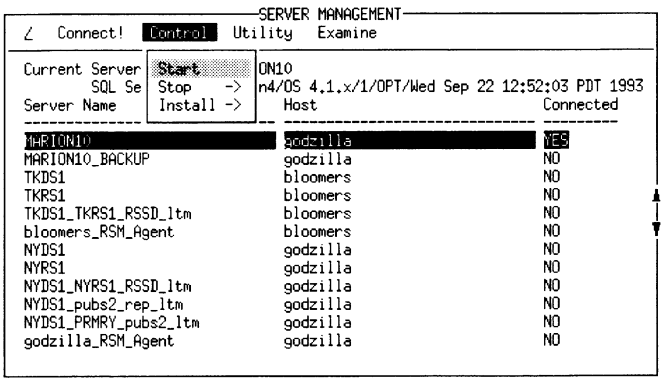
Connecting to a SQL Server

To connect you can either

- Choose Connect! from the menu bar or
- Choose Utility-->Login

SYBASE

Installing, Starting, Stopping a SQL Server



- You must be logged into the OS under the "sybase" account to start or install a SQL Server
- To stop a SQL Server you must be logged in to SA Companion under an account assigned the System Administrator role

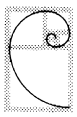
Equivalent Commands

`sybinit`

`startserver`

`shutdown`

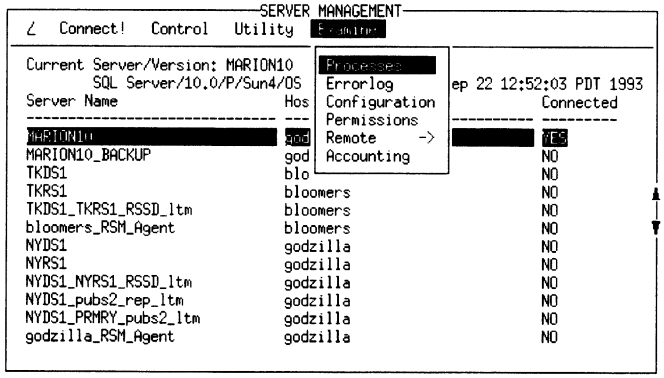
SA Companion can only do the install, start, and stop portions of **sybinit**. **sybinit** can perform much more than the this.



SYBASE

SA Companion

Examining Processes



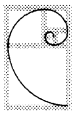
- SA Companion allows you to examine processes for a specific SQL Server
  - Shows locking information for the processes
  - Allows you to stop processes

Equivalent Commands

```
sp_who
sp_lock
kill
```

Examining Processes

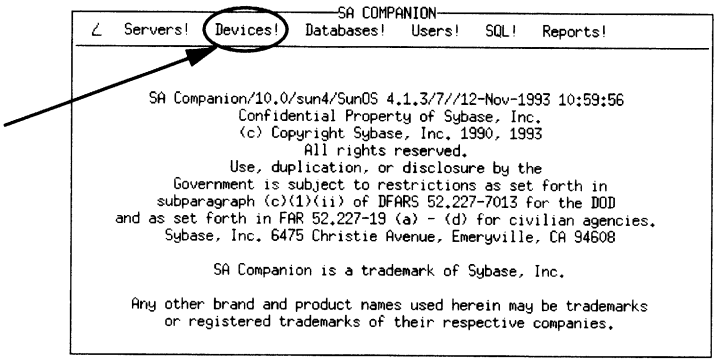
You can do periodic automatic refreshes of both processes and lock information to monitor the SQL Server. You can set up SA Companion to periodically execute, for example, **sp_lock**, by setting a time interval for refreshing the information on screen.

SYBASE 

Other Server Operations

- Run dbcc (dbcc checkdb, dbcc checktable, dbcc checkcatalog, dbcc checkalloc)
- Manage the interfaces file
- Examine the error log
- Configure the server (sp_configure)
- Manage server-level permissions (grant, revoke)
- Examine remote users and servers (requires SA Companion utility be installed on remote machines)
- Examine accounting (sp_monitor)

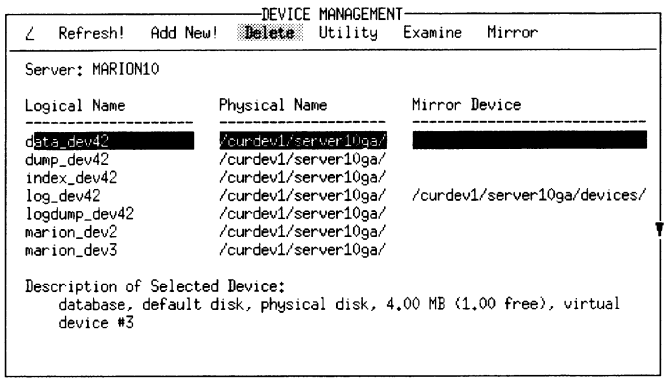
Managing Devices in SA Companion



- The next several slides examine operations you can perform for devices that are configured for a SQL Server installation
- Choose Devices! from the main menu

SYBASE

Adding/Deleting Devices



- All devices are shown for a given SQL Server
- To add a device, choose Add New! from the main menu
- To delete a device, select the device from the list then choose Delete from the main menu

Deleting a Device

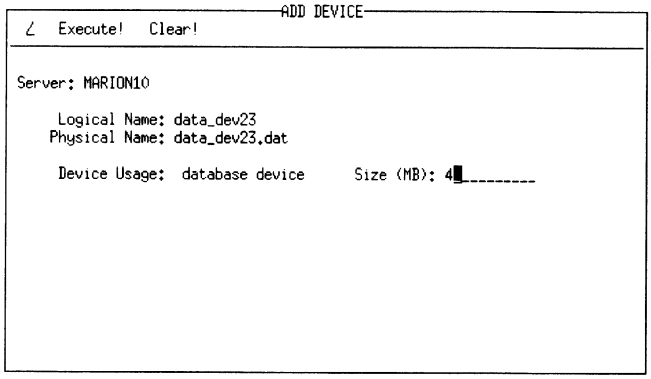
SA Companion will only allow you to delete devices that are unused by any database.

You must restart SQL Server after you drop a device, because the kernel has a process that is accessing the dropped device, and there is no way to kill the process. Restarting frees up the logical device number. Restart with **startserver** or **dataserver**.

Equivalent Commands

```
disk init
sp_dropdevice
```

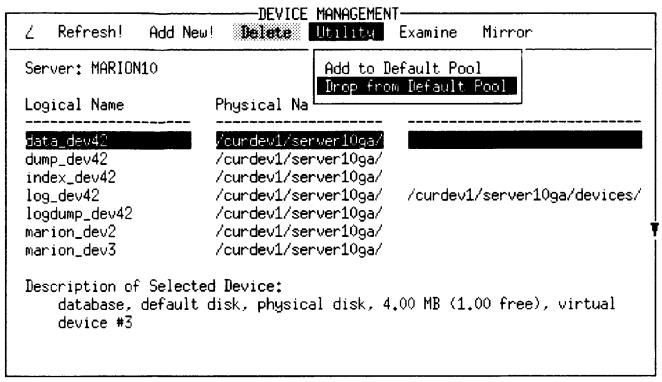
Adding a Device (continued)



- Enter device information into the fields then choose Execute!
- You can add device files or physical devices
  - Include the path for the device

SYBASE

Adding/Dropping a Device from Default Pool

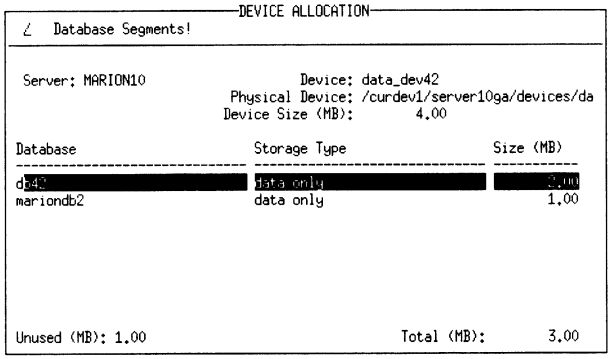


- For a selected device, you can add or drop the device from the default pool
- You can manage device mirroring from this form

Equivalent Command

```
sp_diskdefault
disk mirror
disk unmirror
```

Examining Allocation

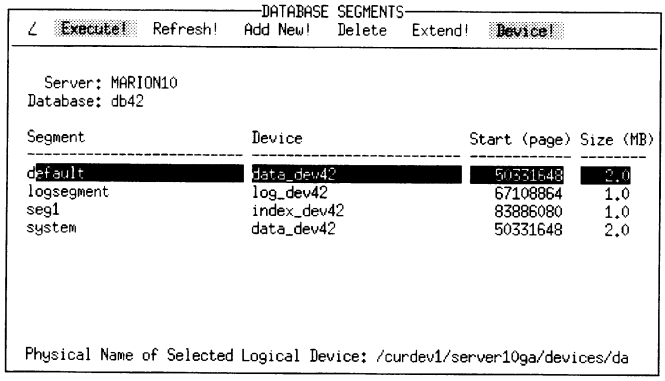


- Choose Examine-->Allocation from the Device Management menu to view this screen
- Database information allocated to the device is shown
- Choose Database Segments! to view segments of a selected database

Equivalent Information View of information from sysdatabases, sysdevices, and sysusages

SYBASE

Manipulating Segments



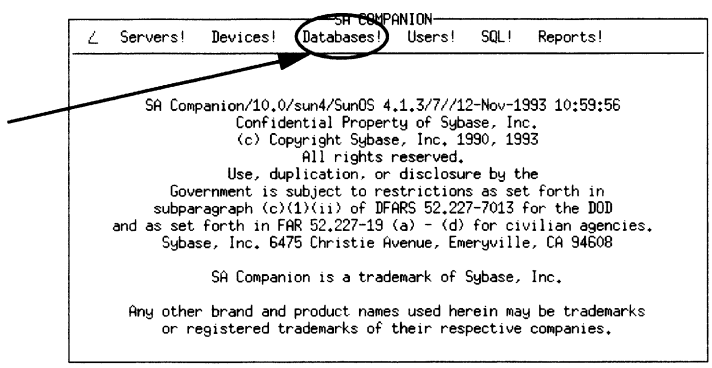
- All segments associated with a database are shown
- You can add, delete (drop) and extend segments from this form

Equivalent Information View of syssegments, sysusages, and sysdevices

Equivalent Commands

```
sp_addsegment
sp_dropsegment
sp_extendsegment
```

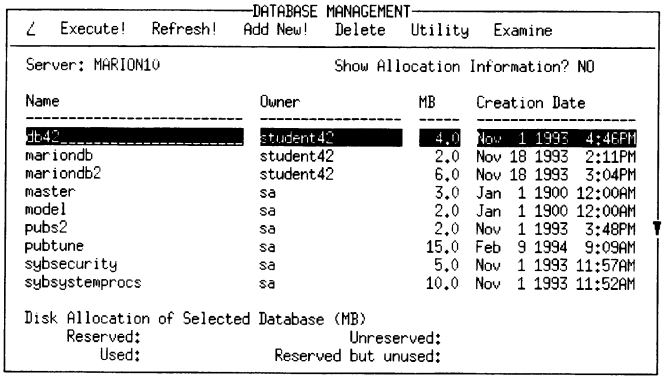
Database Operations in SA Companion



- The next several slides examine the various database operations that can be performed with SA Companion
- Choose Databases! from the main menu

SYBASE

Managing Databases in the SQL Server



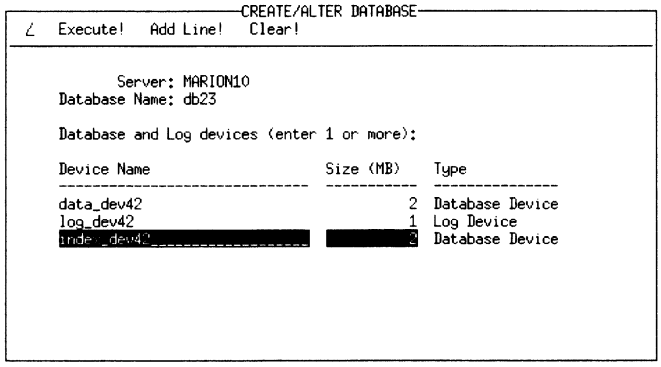
- In the Database Management form you can drop and rename a database
- You can also change the dbo

Equivalent Information View of sysdatabases, syslogins, and sysusages.

Equivalent Command

```
sp_helpdb
sp_renamedb
drop database
sp_changedbowner
```

Creating a New Database

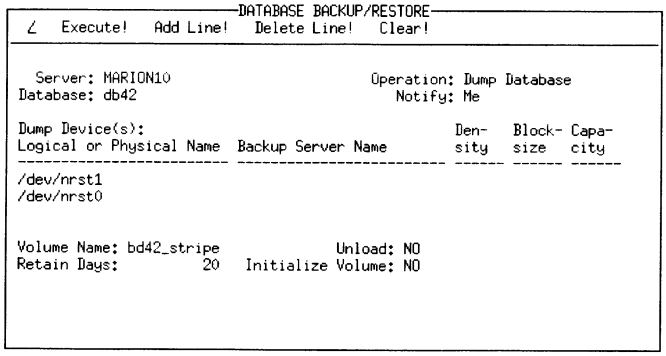


- Choose Add New! from the Database Management form to get to this form
- Specify the database and log devices
- Leave the device information blank if you want the database allocation to use default devices

Equivalent Command `create database`

SYBASE

Backing Up and Restoring Databases

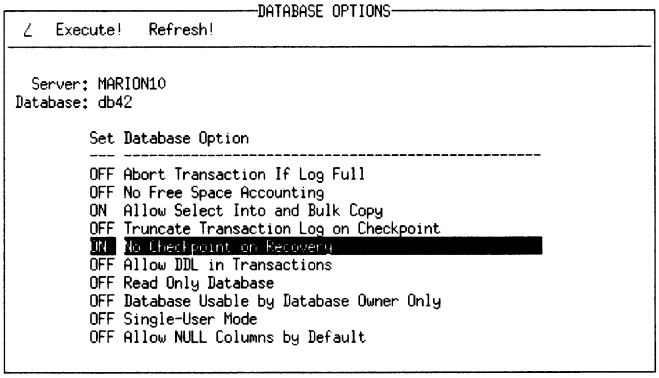


- Choose Utility-->Backup or Utility-->Restore from the Database Management form to display this form
- Select an operation from the Operation rotating choice field
  - Operations are: Dump Database, Dump Transaction Log, Load Database, Load Transaction Log, Dump Tran/No Log, Dump Tran/Trunc Only, Dump Tran/No Trunc

Equivalent Commands

```
dump transaction
dump database
load database
load transaction
```

Manipulating Database Options

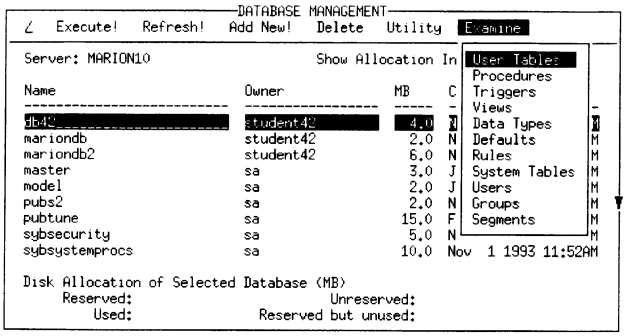


- Choose Utility-->DB Options from the Database Management form to view this form
- View current database option status information and reset options using this form

Equivalent Command `sp_dboption`

SYBASE

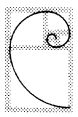
Examining, Creating, and Dropping Database Objects



- From the above pull-down menu you can access all the functions allowing you to manipulate database objects

Equivalent Commands

The equivalent commands encompass all that is required to create, drop, modify and examine database objects, including all the associated **create**, **drop**, and **sp_** stored procedures.



SYBASE

SA Companion

Other Database Management Operations

- Manipulate thresholds (sp_addthreshold, sp_dropthreshold)
- Access the DDL (Data Definition Language) generation utility to:
  - Automatically generate DDL scripts for a 4.9x SQL Server
  - Execute DDL scripts
- Perform space estimation (sp_estspace)

DDL Utility

SA Companion places these scripts in a DDL directory off the SA Companion home directory.

SA Companion currently does not generate DDL scripts for System 10 SQL Servers.

Example List of DDL Files Generated for pubs2 Database

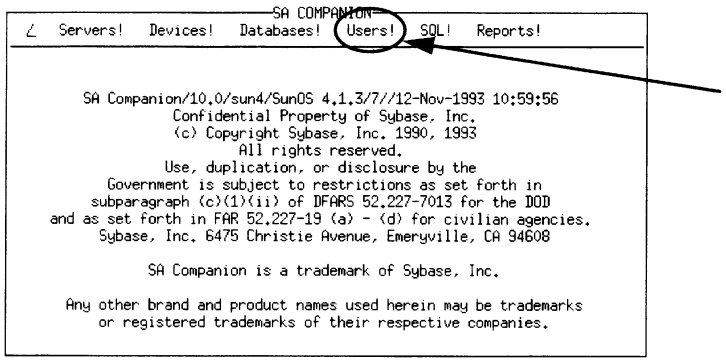
```
aliases.ddl          defaults.ddl
exec.ddl             groups.ddl
indexes.ddl          perms.ddl
procs.ddl            rules.ddl
segments.ddl         tables.ddl
triggers.ddl         types.ddl
users.ddl            views.ddl
```

Partial DDL File Contents--indexes.ddl

```
use pubs2
go

print '    rs_lastcommit_idx'
SETUSER 'dbo'
go
CREATE UNIQUE CLUSTERED INDEX rs_last-
commit_idx
        ON pubs2.dbo.rs_lastcommit (
                origin
        ) ON 'default'

go
SETUSER
go
<etc.>
```

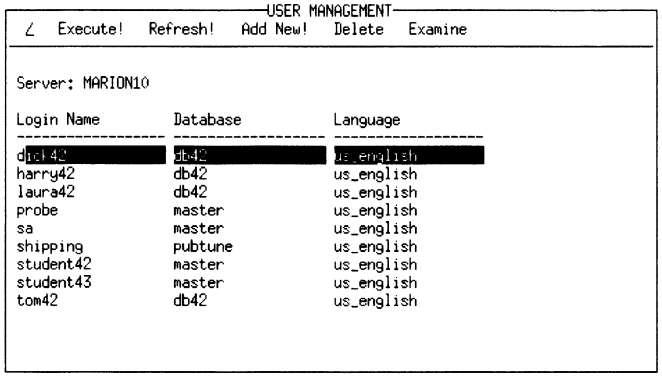
SYBASE

User Operations in SA Companion



- The next several slides examine the various User operations that are performed with SA Companion
- Choose Users! from the main menu

Managing Users



- The User Management form displays all logins for a SQL Server
- Change the login name, default database, and language
- Choose Add New! or Delete to add or drop a login

Equivalent Information View of syslogins

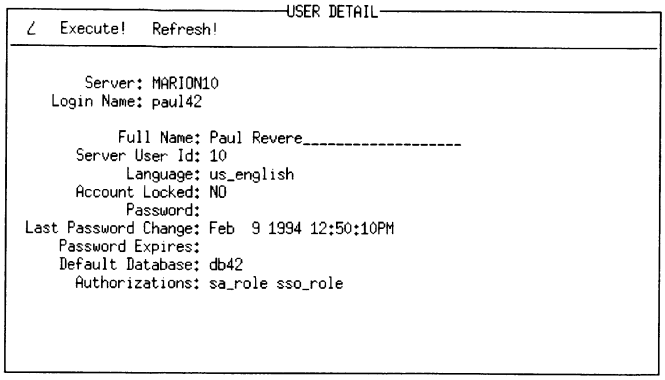
Equivalent Commands

```
sp_modifylogin
sp_addlogin
sp_adduser
sp_dropuser
sp_droplogin
```

SA Companion

SYBASE

Examining and Changing Login Detail



- You can enter/change the following field data: Full Name, Language, Account Locked, Password, Default Database, and Authorizations
- You must have the SSO role in order to enter/change login detail

Equivalent Information View of syslogins, syssrvroles, and sysloginroles

Equivalent Commands

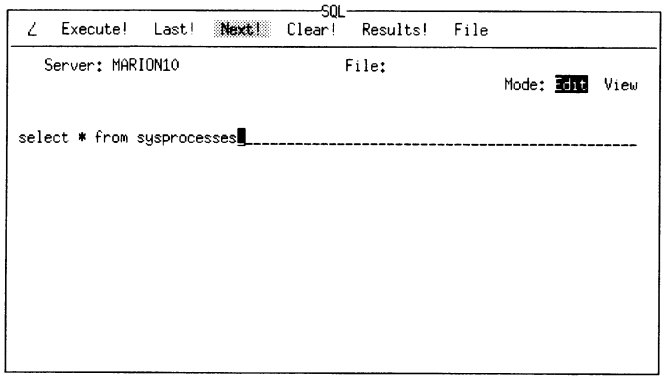
```
sp_displaylogin
sp_helpuser
sp_role
sp_modifylogin
```

Other User Management Operations

- Examine processes for an individual user or all users in the SQL Server (sp_who)
- Examine SQL Server accounting information (sp_monitor)
- Examine objects owned by a user (sp_help)

SYBASE

Submitting SQL to the SQL Server



- Choose SQL! from the main menu to display this form
- SA Companion maintains a history list of all the commands
- SA Companion allows you to save SQL Commands to and retrieve them from a file

Submitting SQL

1. Enter the command in the data area.
2. Choose Execute! to run the query and view the results.

Viewing Results

SQL RESULTS

∠ ⊠ >! Query! File

File:

spid	kpid	enginenum	status	suid	hostname	progra
2	327685	0	sleeping	0		
3	393222	0	sleeping	0		
4	458759	0	sleeping	0		
5	524296	0	sleeping	0		
6	2555943	0	running	1	beast	SAC
10	13041675	0	send sleep	1	ib	isql

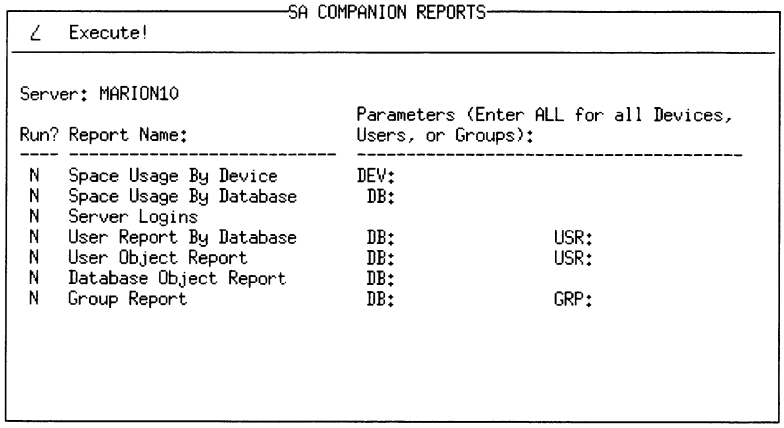
6 rows returned

- This is the output from the SQL statement `select * from sysprocesses` shown in the previous slide
- Scroll through the results using the scroll arrows in the menu bar and on the window, if present

SA Companion

SYBASE

Generating Reports Using SA Companion



- Choose Reports! from the main menu to display this form
- You must specify the database for reports requesting the database name (ALL is not supported)

Generating Reports

Reports are written to an ASCII file with a **.lis** suffix in the current directory, the directory from which SA Companion was started.

Example Report Headers

Server Logins Report Headers: User Name (privileges), Server user Id, Default Database, Creation Time

Space Usage By Database Report Headers: Database Name, Owner, Dbid, Created, Space Usage

Database Object Report Headers: Object Type, Object Name (privilege), Object Id, Creation Date, Owner Name

Summary

- SA Companion allows you to efficiently perform administration tasks on any 4.9x or 10.0 SQL Server
  - Knowledge of command syntax is not required
  - Focus on what you are doing, not on what the SQL Server is doing internally
  - Appropriate roles and permissions is required for the administration tasks
- SA Companion allows you to work with local and remote SQL Servers

Lab 11a – Using SA Companion

You will use SA Companion to view SQL Server and database information in the SQL Server installed for the class. You will also add a user to your database.

You will have the opportunity to create, modify, and delete objects and devices with SA Companion in the Case Study Module.

1. Log in to SA Companion using your studentNN identification.

2. Examine the processes running on the Server.

3. Database Operations:
   a. View your database options (dbNN).
   b. Examine the various database objects for your database (tables, procedures, data types, system tables, users, groups, segments, etc.)

4. Device Operations:
   a. View the devices for the SQL Server.
   b. Examine the allocation for the devices in your database.

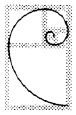
5. User Operations:
   a. Examine the processes for your login.
   b. Examine the objects owned and accounting for your login.
   c. Add a new login for your database. After adding the login, give this login sa_role.

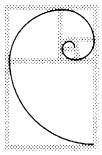
6. Submit some SQL to the SQL Server. For example:
   a. sp_who, sp_helpdb, sp_help
   b. select * from sysusers

7. Run some of the reports supplied by SA Companion. View the reports from your current directory. The reports have a **.lis** suffix. After running the "Space Usage By Database" report, for example, from the UNIX prompt in your current directory you would type:

   ```
   more db.lis,m
   ```

   then press **Return** to view the report.





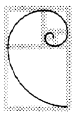
SYBASE®

Module 12

System Administration Case Study

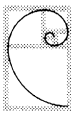
Objectives

- Practice skills acquired in the previous modules of the course in a near "real-world" scenario
- Troubleshoot common systems administration problems in a realistic scenario, given a statement of the problem and/or a system error message

SYBASE 

Case Study Scenario

- Company XYZ:
  - Is a rapidly growing company
  - Is a current satisfied user of SQL Server
  - Needs to migrate a database from a previous Server installation to a new one
- Installation of the new SQL Server and the database migration must be performed efficiently
  - Planning is important
- Previous SQL Server installation needs to be cleaned up
- You are responsible for performing this task, both its planning and its implementation



SYBASE

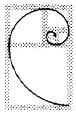
System Administration Case Study

Planning for Migration of a Database

- Preparing the database for movement to another SQL Server
- Backing up the database
- Installing the new SQL Server
- Reconfiguring the system
- Getting database and device configuration information
- Loading the new database
- Validating all activities along the way
- Dropping resources and information from previous SQL Server installation
- And more!

Planning for Migration

Part of your task is to plan for the migration of the database to a new SQL Server installation, and to define each step you would take.

SYBASE 

Doing the Case Study

- Write up the solutions to the case study, then perform the tasks in the order specified
- The instructor will perform a SQL Server installation and reconfiguration as a class activity
  - Do not perform the install individually
- Check the module answer key:
  - After completing the plan
  - Before executing commands for each task in the plan
  - For additional information
- Upon completing the case study, perform the "mini-case studies" that follow it for additional practice on common administrative tasks

Lab 12a: Practicing System Administration Skills

Case Study Scenario

Because of the growth of company XYZ, you will have to migrate your database from a previous SQL Server installation to a new one. After backing up the database (dbNN), you will install a new SQL Server named "PROD2", then load the database onto the new installation into a database named "deptNN". After verifying that this has been successfully achieved, perform the necessary house-cleaning on the original server and its associated devices and resources.

The new SQL Server installation:

- Accepts the default values for installation of the SQL Server with the exceptions below
- Recognizes a maximum of:
  - 90 devices
  - 100 users
  - 25 open databases
- Requires configuration/set-up so that database structure/configuration in the new installation will allow the database being loaded to correctly place objects on devices and segments.

1. The first order of priority is to plan. Write down the tasks you will need to perform in order to successfully migrate the database. Keep in mind that all the data you have is available from the current SQL Server install with your database.

(over)

2. Check your plan against the plan in the answer key. Make any appropriate changes. Write out the commands required for each task in the plan in the space provided below. Completely write out the commands before actually implementing the plan. **Note**: Plan steps below are those indicated in the answer key. If you wish to revise these steps based on your plan, please do so and check with the instructor.

   a. Gather and record information about the database being moved in order to re-create it correctly in the new environment.

   b. Verify the integrity of the database.

   c. Dump the database, then verify the dump.

   d. Install the SQL Server, PROD2 (instructor demonstration).

   e. Reconfigure Server to handle the defined number of devices, users, and open databases—90, 100, and 25 respectively (instructor demonstration).

   f. Shutdown and restart the Server to activate the new configuration options (instructor demonstration).

g. Add the login and roles for the login of the user who is dbo of the **deptNN** database. Grant dbo **create database** permission.

h. Log in as dbo then create the appropriate devices that will be used by database **deptNN**. Note: Use the same device numbers as you did earlier in the class (see Lab 3a).

i. Create the database **deptNN** according to the information gathered in 2.a.

j. Load the backed up database into the newly created database and confirm successful loading of the database into the appropriate devices.

k. Set up the remaining logins, users, and permissions (describe how you would implement this--you will not actually perform this on the Server).

1. Drop the database **dbNN** and its devices from the previous SQL Server installation.

3. Execute the above tasks on the SQL Server.

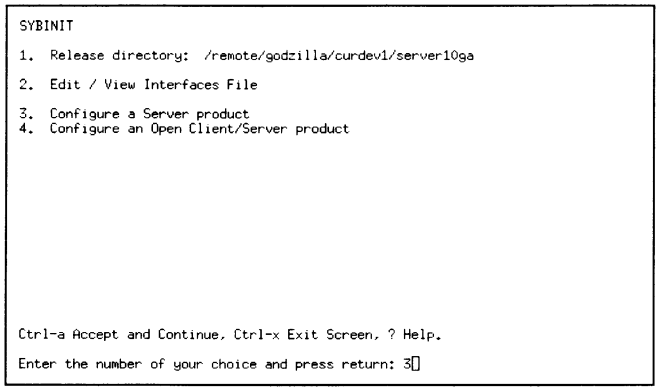
4. If time permits, continue to the mini-case studies in Lab 13b, following the SQL Server installation session in the next slides.

Using sybinit to Install a SQL Server

- The sybinit utility is in the install subdirectory of the SQL Server
- Move to the install subdirectory and type sybinit to begin the installation

SYBASE

Installing the SQL Server



- The SYBASE variable, or your platform equivalent, points to the release directory
- The first step in installation is to "Configure a Server product"

Installing the SQL Server This installation session is on a UNIX system

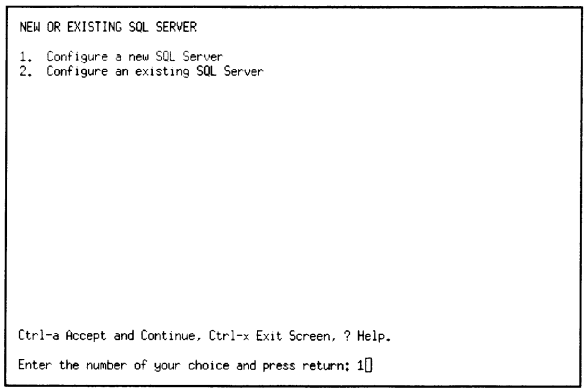
Configuring a Server Product

```
CONFIGURE SERVER PRODUCTS

Products:

    Product                 Date Installed    Date Configured
1.  SQL Server              Sep 20 93 16:35   Nov 01 93 11:58
2.  Backup Server           Sep 20 93 16:35   Feb 09 94 14:28









Ctrl-a Accept and Continue. Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

- For an initial installation, you must configure both a SQL Server and a Backup Server
- Choose **1** to configure a SQL Server product

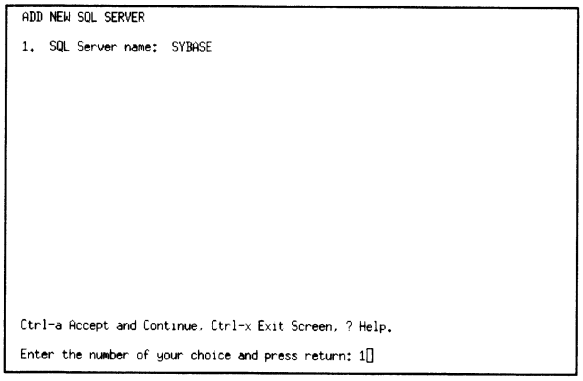
SYBASE

Configuring a New SQL Server

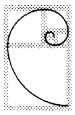


- Choose **1** to configure a new SQL Server, then press **Return**

Configuring a New SQL Server (continued)



- Choose **1** to change the SQL Server name, then press **Return**

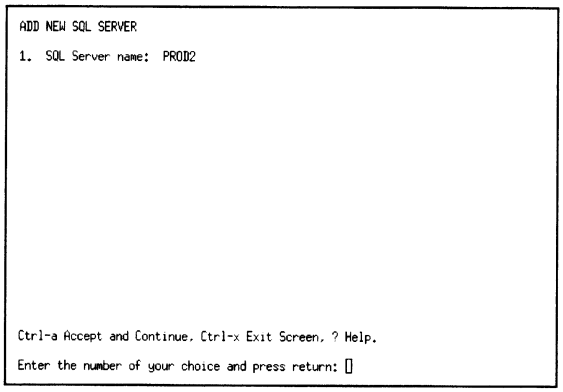
SYBASE 

Configuring a New SQL Server (continued)

```
1.  SQL Server name:  SYBASE




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
Enter the name of the new SQL Server (default is 'SYBASE'):
PROD2
```

- Enter **PROD2** as the new SQL Server name, then press **Return**

Configuring a New SQL Server (continued)



- Press **Ctrl+a** to accept the new name of the SQL Server

SYBASE

Configuring a New SQL Server (continued)

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY              Incomplete

2.  MASTER DEVICE CONFIGURATION                          Incomplete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                Incomplete
4.  SET ERRORLOG LOCATION                                Incomplete
5.  CONFIGURE DEFAULT BACKUP SERVER                      Incomplete

6.  CONFIGURE LANGUAGES                                  Incomplete
7.  CONFIGURE CHARACTER SETS                             Incomplete
8.  CONFIGURE SORT ORDER                                 Incomplete

9.  ACTIVATE AUDITING                                    Incomplete




Ctrl-a Accept and Continue. Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

- The main SQL Server configuration menu appears:
  - All items must be marked "Complete" to install the SQL Server
  - For simplicity, select the items in the order they appear
- Choose **1** to configure the interfaces file

Configuring the Interfaces File

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:  PROD2

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 3
```

- Choose **3** to define the listener service for the SQL Server
  - The listener service defines the location of the SQL Server on the network

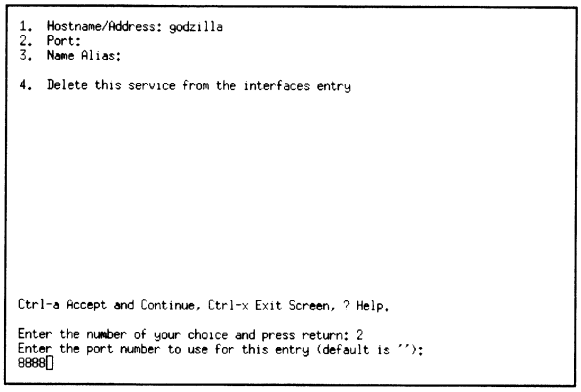
SYBASE

Configuring the Interfaces File (continued)

```
EDIT TCP SERVICE

1.  Hostname/Address: godzilla
2.  Port:
3.  Name Alias:

4.  Delete this service from the interfaces entry









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
```

- A listener service requires at least the Hostname/Address and Port
  - Accept the supplied hostname/address
  - Enter a port but no name alias
- Choose **2** to enter the Port number

Configuring the Interfaces File (continued)



- Enter the port number **8888** for this installation, then press **Return**

SYBASE

Configuring the Interfaces File (continued)

```
EDIT TCP SERVICE

1.  Hostname/Address: godzilla
2.  Port: 8888
3.  Name Alias:

4.  Delete this service from the interfaces entry









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- Press **Ctrl+a** to accept the entry
  - After accepting, confirm the information by pressing **Y**

Configuring the Interfaces File (continued)

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:  PROD2

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service

Modify or delete a service

Listener services available:

    Protocol  Address          Port          Name Alias
4.  tcp       godzilla         8888


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- The Server Interfaces File Entry Screen summarizes what has been entered
  - Press **Ctrl+a** to accept the information, then press **Y** to write the information to the interfaces file

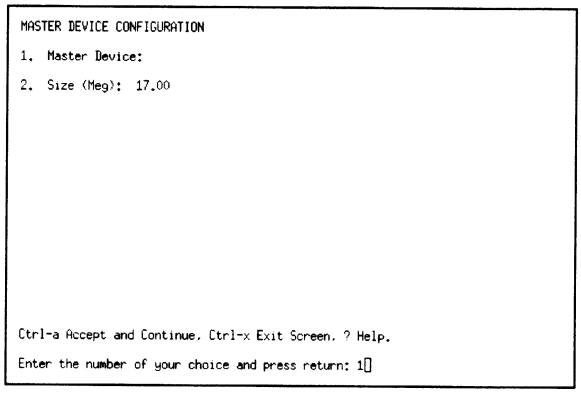
SYBASE

Configuring the Master Device

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                           Incomplete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                 Incomplete
4.  SET ERRORLOG LOCATION                                 Incomplete
5.  CONFIGURE DEFAULT BACKUP SERVER                       Incomplete

6.  CONFIGURE LANGUAGES                                   Incomplete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
```

- You return to the SQL Server configuration screen after writing the new interfaces file entry for your SQL Server installation
  - Note that item 1 is marked "Complete"
- Choose **2** to configure the master device, then press **Return**

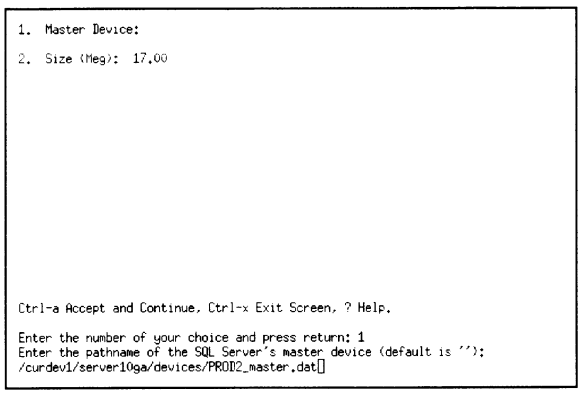
Configuring the Master Device (continued)



- In this screen, specify the master device location and the master device size
  - The default size of the master device is 17 Megabytes
- Choose **1** to enter the master device name and location

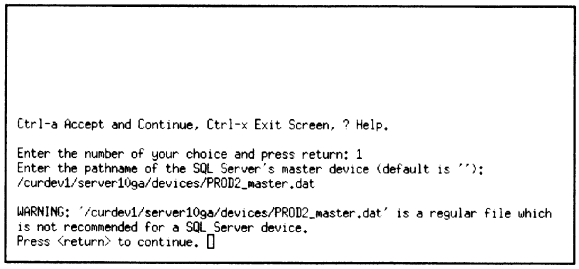
SYBASE

Configuring the Master Device (continued)



- Enter the path and device name for the master device, then press **Return**
  - In this example, the master device is a file

Configuring the Master Device (continued)



- On UNIX systems, Sybase recommends that the master device NOT be a file
- A warning is displayed
- Press **Return** to continue

SYBASE

Configuring the Master Device (continued)

```
MASTER DEVICE CONFIGURATION

1.  Master Device:  /curdev1/server10ga/devices/PROD2_master.dat

2.  Size (Meg):  17.00









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- The master device configuration data is displayed
  - Press **Ctrl+a** to accept the data

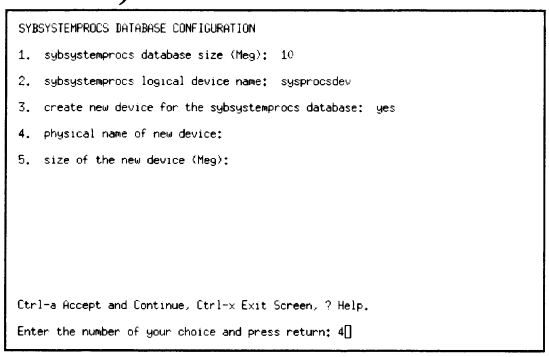
Configuring sybsystemprocs Database

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                             Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                   Incomplete
4.  SET ERRORLOG LOCATION                                   Incomplete
5.  CONFIGURE DEFAULT BACKUP SERVER                         Incomplete

6.  CONFIGURE LANGUAGES                                     Incomplete
7.  CONFIGURE CHARACTER SETS                                Incomplete
8.  CONFIGURE SORT ORDER                                    Incomplete

9.  ACTIVATE AUDITING                                       Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen. ? Help.

Enter the number of your choice and press return: 3
```

- Choose **3** to configure the **sybsystemprocs** database

SYBASE

Configuring sybsystemprocs Database (continued)



- Enter the physical name of the new device and its size, **/curdev1/server10ga/devices/PROD2_procs.dat** and **10**, respectively
- If you do not enter a new device, **sybsystemprocs** is placed on the master device

Configuring sybsystemprocs

For item 3, "yes" is default and is recommended.

If you answer "no" to item 3, and have configured your master device large enough, sybsystemprocs is placed on the master device.

Configuring sybsystemprocs Database (continued)

```
SYBSYSTEMPROCS DATABASE CONFIGURATION

1.  sybsystemprocs database size (Meg):  10

2.  sybsystemprocs logical device name:  sysprocsdev

3.  create new device for the sybsystemprocs database:  yes

4.  physical name of new device:  /curdev1/server10ga/devices/PROD2_procs.dat

5.  size of the new device (Meg):  10









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

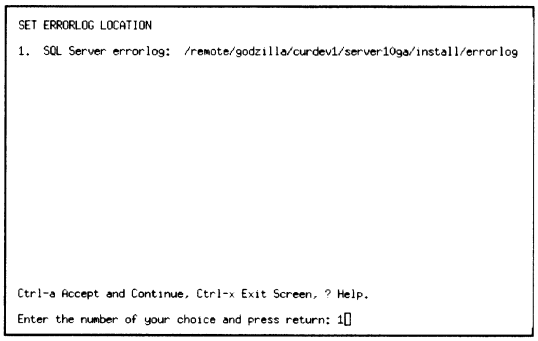
- Press **Ctrl+a** to accept the configuration information and to return to the SQL Server Configuration menu

Setting the Error Log Location

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                            Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                  Complete
4.  SET ERRORLOG LOCATION                                 Incomplete
5.  CONFIGURE DEFAULT BACKUP SERVER                       Incomplete

6.  CONFIGURE LANGUAGES                                   Incomplete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 4
```

- Choose **4**, then press **Return** to set the error log location

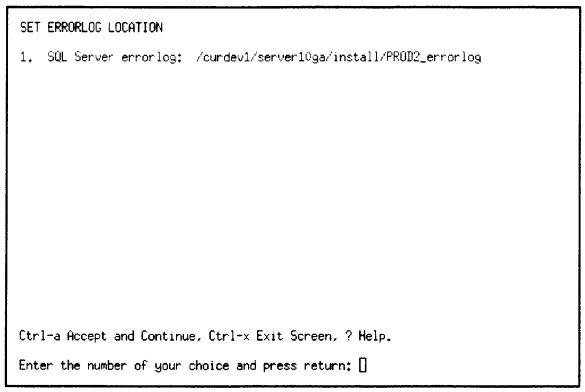
Setting the Error Log Location (continued)



- The default name for the error log is "errorlog," and it is placed in the install directory of the SQL Server
  - Choose **1** to modify the name of the error log
  - Type in the path and name of the error log file **/curdev1/server10ga/install/PROD2_errorlog**, then press **Return**

SYBASE

Setting the Error Log Location (continued)



- Press **Ctrl+a** to accept the error log setup information

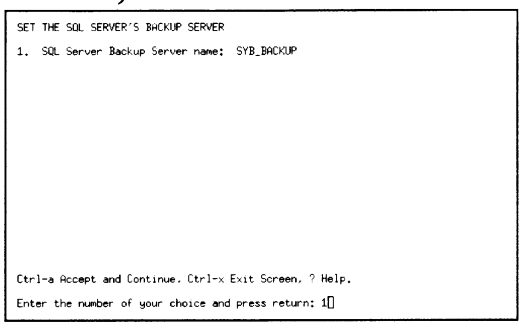
Configuring the Default Backup Server

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                             Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                   Complete
4.  SET ERRORLOG LOCATION                                   Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                       Incomplete

6.  CONFIGURE LANGUAGES                                   Incomplete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 5
```

- Choose **5** to configure the default Backup Server from the SQL Server Configuration screen

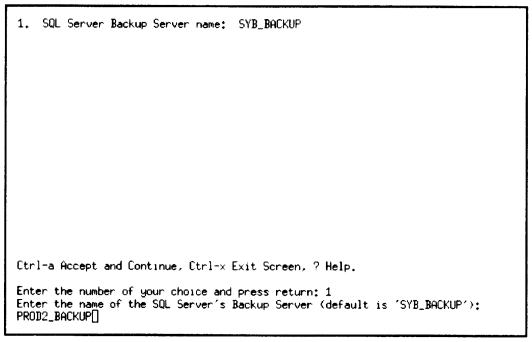
SYBASE

Configuring the Default Backup Server (continued)



- Specify the name of the default Backup Server this SQL Server installation recognizes (SYB_BACKUP is the default name)
- A separate configuration for the host and port of the Backup Server is performed later
- Choose **1** to change the name of the Backup Server

Configuring the Default Backup Server (continued)



- Enter the name, then press **Return**

SYBASE

Configuring the Default Backup Server (continued)

```
SET THE SQL SERVER'S BACKUP SERVER

1.  SQL Server Backup Server name:  PROD2_BACKUP









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- Press **Ctrl+a** to accept the new Backup Server name and to return to the SQL Server Configuration Menu

Configuring Languages

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                            Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                  Complete
4.  SET ERRORLOG LOCATION                                  Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                        Complete

6.  CONFIGURE LANGUAGES                                   Incomplete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue. Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 6
```

- Choose **6** to configure the language(s) supported by the SQL Server installation

SYBASE

Configuring Languages (continued)

```
CONFIGURE LANGUAGES

    Current default language:  us_english
    Current default character set:  ISO 8859-1 (Latin-1) - Western European
8-bit character set.
    Current sort order:  Binary ordering, for the ISO 8859/1 or Latin-1
character set (iso_1).

Select the language you want to install, remove, or designate as the default
language.

    Language                Installed?  Remove      Install     Make default
1.  us_english              yes         no          no          yes




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- You can install multiple languages for a SQL Server installation
- Press **Ctrl+a** to accept the default language and parameters of this screen

Configuring Character Sets and Sort Order

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                             Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                   Complete
4.  SET ERRORLOG LOCATION                                   Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                         Complete

6.  CONFIGURE LANGUAGES                                     Complete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 7
```

- Choose **7** to configure character sets--accept the defaults
- Choose **8** to configure sort order--accept the defaults

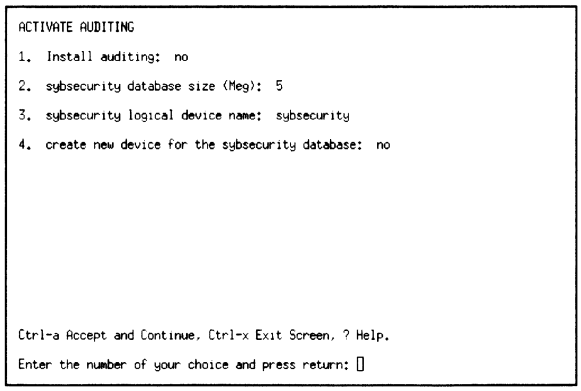
SYBASE

Activating Auditing

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                            Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                  Complete
4.  SET ERRORLOG LOCATION                                  Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                        Complete

6.  CONFIGURE LANGUAGES                                    Complete
7.  CONFIGURE CHARACTER SETS                               Complete
8.  CONFIGURE SORT ORDER                                   Complete

9.  ACTIVATE AUDITING                                     Incomplete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 9
```

- Choose **9** to activate/deactivate auditing during installation
  - You must choose this option even if you are not planning to use the auditing facility

Activating Auditing (continued)



- The default is not to install auditing during SQL Server installation
- Press **Ctrl+a** to accept the default values

SYBASE

Executing the Configuration

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY                Complete

2.  MASTER DEVICE CONFIGURATION                            Complete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                  Complete
4.  SET ERRORLOG LOCATION                                  Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                        Complete

6.  CONFIGURE LANGUAGES                                    Complete
7.  CONFIGURE CHARACTER SETS                               Complete
8.  CONFIGURE SORT ORDER                                   Complete

9.  ACTIVATE AUDITING                                      Complete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- When each item in the menu is marked "Complete," configuration of the SQL Server is completed
- Press **Ctrl+a** to continue with the installation process, then confirm that you want to execute the SQL Server configuration

Executing the Configuration (continued)

```
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                    Complete
4.  SET ERRORLOG LOCATION                                    Complete
5.  CONFIGURE DEFAULT BACKUP SERVER                          Complete

6.  CONFIGURE LANGUAGES                                      Complete
7.  CONFIGURE CHARACTER SETS                                 Complete
8.  CONFIGURE SORT ORDER                                     Complete

9.  ACTIVATE AUDITING                                        Complete




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
Execute the SQL Server Configuration now? y

WARNING: '/curdev1/server10ga/devices/PROD2_master.dat' is a regular file which
is not recommended for a SQL Server device.
Press <return> to continue.
```

- If you specified a device file for the master database, you are again warned that this is not recommended
  - Press **Return** to continue

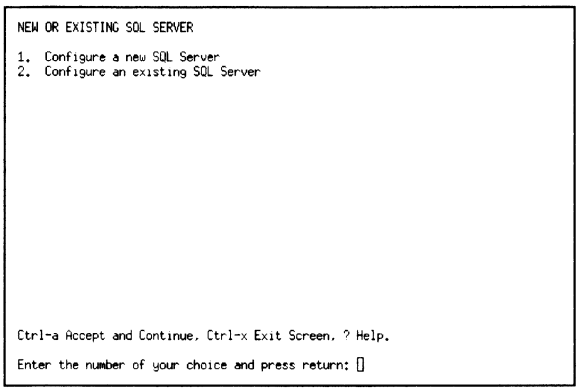
SYBASE

Executing the Configuration (continued)

```
Running task to install system stored procedures.
.....
.....Done
Task to install system stored procedures succeeded.
Running task to set permissions for the 'model' database.
.Done
Task to set permissions for the 'model' database succeeded.
Running task to set the Backup Server for the SQL Server.
Task to set the Backup Server for the SQL Server succeeded.
Running task to set the default character set and/or default sort order for the
SQL Server.
Setting the default character set to iso_1
Sort order 'binary' has already been installed.
Character set 'iso_1' is already the default.
Sort order 'binary' is already the default.
Task to set the default character set and/or default sort order for the SQL
Server succeeded.
Running task to set the default language.
Setting the default language to us_english
Language 'us_english' is already the default.
Task to set the default language succeeded.

Configuration completed successfully.
Press <return> to continue.
```

- Messages are echoed onto the screen during the installation process
- Wait until the message "Configuration completed successfully" appears
  - Press **Return** to continue

Configuring the Backup Server



- After the installation of the new SQL Server, you return to this menu
- Press **Ctrl+a** to return to the previous menu that allows you to configure a Backup Server

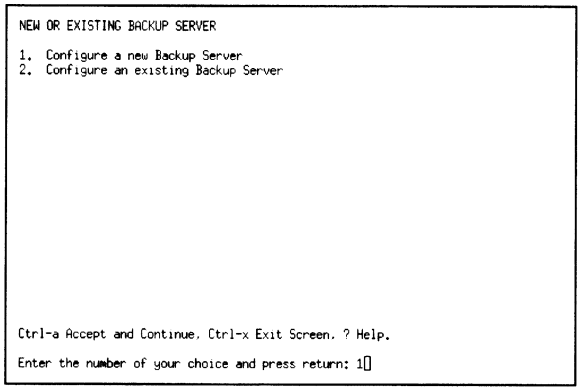
SYBASE

Configuring the Backup Server (continued)

```
CONFIGURE SERVER PRODUCTS

Products:

    Product                 Date Installed    Date Configured
1.  SQL Server              Sep 20 93 16:35   Nov 01 93 11:58
2.  Backup Server           Sep 20 93 16:35   Feb 09 94 14:28









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
```

- Choose **2** to configure the Backup Server that the SQL Server will recognize

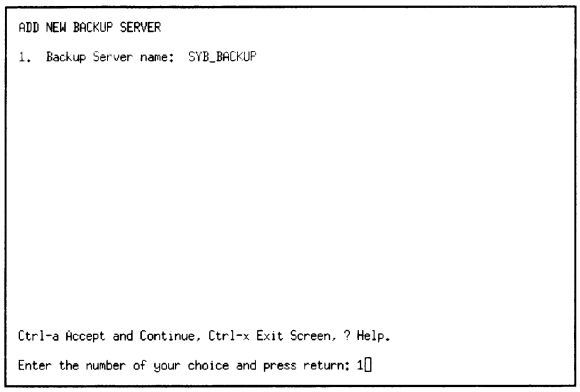
Configuring the Backup Server (continued)



- Choose **1** to configure a new Backup Server
- Choose **2** to change any details for a Backup Server already installed on this system

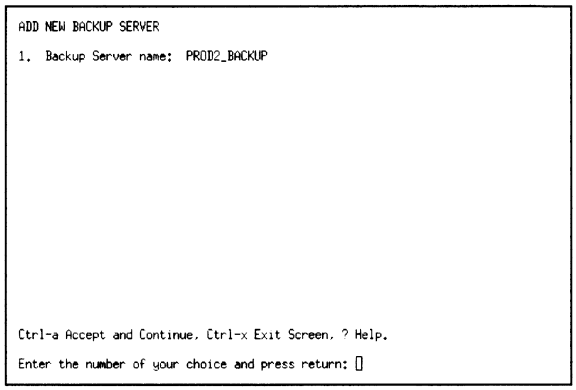
SYBASE

Configuring the Backup Server (continued)

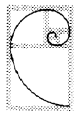


- Choose **1** to change the name of the backup server
  - Enter the name **PROD2_BACKUP**, then press **Return**

Configuring the Backup Server (continued)



- Press **Ctrl+a** to accept the changes and continue

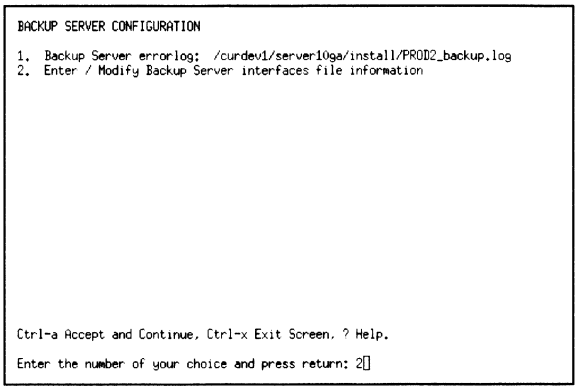
SYBASE 

Configuring the Backup Server (continued)

```
BACKUP SERVER CONFIGURATION

1.  Backup Server errorlog:  /remote/godzilla/curdev1/server10ga/install/backup
2.  Enter / Modify Backup Server interfaces file information









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

- Choose **1** to change the name and/or location of the Backup Server error log
  - The name and location are **/curdev1/server10ga/install/PROD2_backup.log**

Configuring the Backup Server (continued)



- Choose **2** to enter interfaces file information for the Backup Server

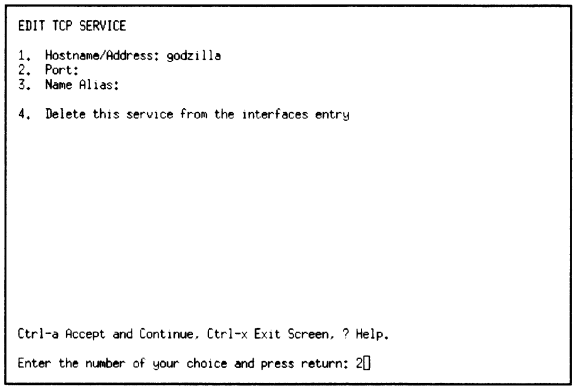
SYBASE

Configuring the Backup Server (continued)

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:  PROD2_BACKUP

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service









Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 3
```

- Choose **3** to add a new listener service for the Backup Server

Configuring the Backup Server (continued)



- Choose **2** to enter the Backup Server port
- The hostname is correct
- No name alias will be used for the Backup Server

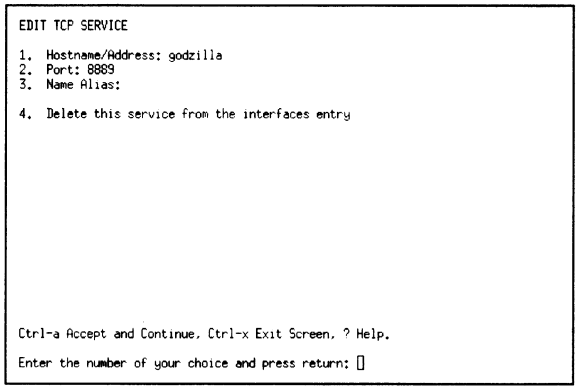
SYBASE

Configuring the Backup Server (continued)

```
1.  Hostname/Address: godzilla
2.  Port:
3.  Name Alias:

4.  Delete this service from the interfaces entry




Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
Enter the port number to use for this entry (default is ''):
8889
```

- Enter the port number, then press **Return**
  - The same criteria apply as to the SQL Server

Configuring the Backup Server (continued)



- Press **Ctrl+a** to accept the Backup Server listener service information

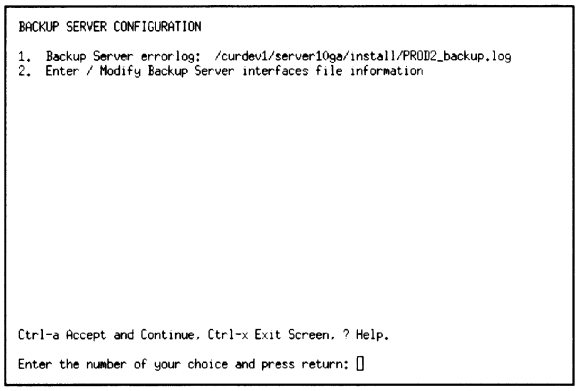
SYBASE

Configuring the Backup Server (continued)

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:  PROD2_BACKUP

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service

Modify or delete a service

Listener services available:

    Protocol  Address          Port         Name Alias
4.  tcp       godzilla         8889


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: []
```

- Press **Ctrl+a** to accept the Backup Server interfaces file information
- Confirm that you want to write these changes to the interfaces file

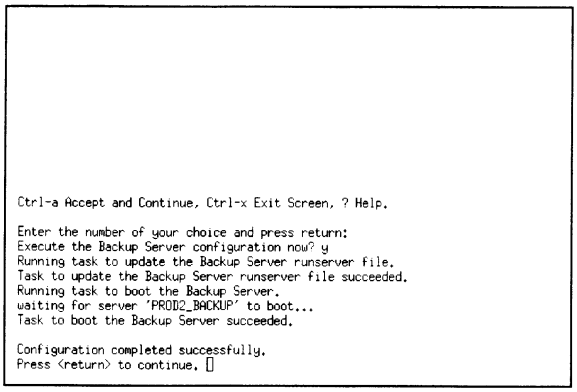
Configuring the Backup Server (continued)



- Press **Ctrl+a** to accept the Backup Server configuration, then confirm that you want to execute the Backup Server configuration

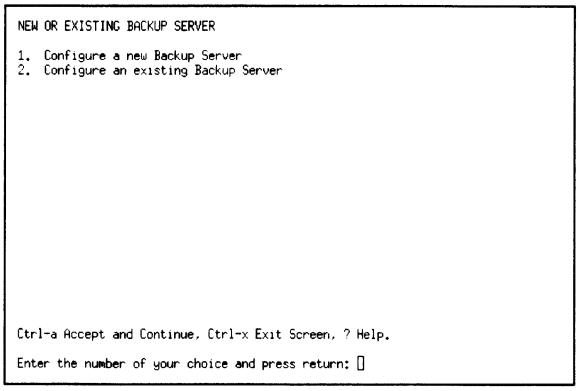
SYBASE

Executing the Backup Server Configuration



- The message "Configuration completed successfully" appears after the Backup Server is installed and configured
- Press **Return** to continue

Installation Complete



- From this point, back out to the main menu, then exit **sybinit**
- Press **Ctrl+a** or **Ctrl+x** to continue until you return to the main menu

SYBASE

Exiting sybinit

```
2.  Edit / View Interfaces File

3.  Configure a Server product
4.  Configure an Open Client/Server product









Ctrl-a Accept and Continue. Ctrl-x Exit Screen. ? Help.

Enter the number of your choice and press return:

Exiting.
The log file for this session is '/curdev1/server10ga/init/logs/log0214.005'.
godzilla% []
```

- Press **Ctrl+a** or **Ctrl+x** to exit the main menu screen
- A log file for the sybinit session is written for your reference

Lab 12b: Mini-Case Studies: Solving Common Systems Administration Problems (optional)

The following are paper-based activities, with the exception of exercise 15. Write your answers in the spaces provided. You may wish to confirm some of the commands you propose as solutions to the scenarios on the SQL Server.

RECOVERABILITY AND MIRRORING

1. You have moved a database from development to production. Requirements for the production database are:
   - data and log are mirrored separately
   - database is recoverable from transaction log dumps

   Examine the output below.

```
1> sp_helpdb devel42
2> go
 name                 db_size       owner                  dbid
        created
        status

 -------------------- ------------ ---------------------- ------
       --------------

------------------------------------------------------------------------
------------------------------
 devel42                  15.0 MB sa                           8
        Feb 15, 1994
        trunc log on chkpt

 device_fragments             size          usage                free kbytes
 ---------------------------- ------------ -------------------- -----------
 devel_dev                    15.0 MB       data and log               14672
(return status = 0)
```

Is the database configured correctly to satisfy the production requirements? If not, what is the problem(s) and how will it be corrected?

SEPARATING DATABASES ON A SHARED DEVICE

2. There are two databases, **db1** and **db2**, on a single server. Both databases have their data segments on a device called **data_dev1**, and their logs on a device called **log_dev1**. To accommodate growth in the databases, you need to move **db2**'s data segments to a new device. What steps do you take to achieve this?

STANDARDIZING DATABASE CONFIGURATION

3. Your organization has standard procedures, rules, defaults, user-defined datatypes, and database options that would be used for most databases. How would you streamline the incorporation of all these objects and options into future databases you create?

WORKING WITH SEGMENTS

4. You have a large table in your database, **dept42**, that you want to split evenly across two devices (**tab_dev1** and **tab_dev2**). You already have the ability to load the data in two portions. What procedure would you use? You will be using the following stored procedures: sp_addsegment, sp_dropsegment, sp_extendsegment.

MONITORING THE ERROR LOG

5. The login with OPER role performs database and transaction dumps of the production database on a daily basis. As sa, you carefully monitor the error log routinely and find the following information. What should you do?

   Error log excerpt:

```
00:94/02/15 09:49:53.67 server WARNING: **************************
00:94/02/15 09:49:53.81 server Attempt by user 1 to dump xact on db db42
with NO_LOG
00:94/02/15 09:49:54.86 server Attempt by user 1 to dump xact on db db42
with NO_LOG was successful
00:94/02/15 09:49:54.86 server WARNING: **************************
```

6. Users are receiving the following messages at the client application. From the OS prompt, "%", the client sees:

```
% isql -Usa -P -SMARION10
DB-LIBRARY error:
      Unexpected EOF from SQL Server.
%
```

You check the error log and find the following:

```
00:94/02/15 09:52:54.50 server Error: 21, Severity: 21, State: 1
00:94/02/15 09:52:54.53 server WARNING - Fatal Error 1601 occurred at Feb
15 1994 9:52AM. Please note the error and time, and contact a user with
System Administrator (SA) authorization.
00:94/02/15 09:52:54.56 server Error: 1601, Severity: 21, State: 3
00:94/02/15 09:52:54.56 server There are not enough 'user connections'
available to start a new process. Retry when there are fewer active users,
or ask your System Administrator to reconfigure SQL Server with more user
connections.
```

What can you do to prevent this from happening again?

MONITORING SPACE USAGE

7. You have your database and some tables distributed on various devices and segments. One of the users complains that he/she is receiving the following message after attempting an insert on table, **big_tab**:

```
Msg 1105, Level 17, State 1:
Line 1:
```

```
Can't allocate space for object 'big_tab' in database 'dept42' because the
'tab_seg1' segment is full. If you ran out of space in syslogs, dump the
transaction log. Otherwise, use ALTER DATABASE or sp_extendsegment to
increase the size of the segment.
```

What could be the cause of this error, and where would you look to find additional information to validate your target root cause?

8. For the solution to question 7, which commands/stored procedures would you execute to find the information and correct the problem?

THRESHOLDS

9. You have the following **sp_thresholdaction** procedure defined in **sybsystemprocs**. (Note: This procedure is also in the *Systems Administration Guide*.)

```
create procedure sp_thresholdaction
    @dbname                 varchar(30),
    @segmentname                varchar(30),
    @space_left                 int,
    @status                 int
as
declare @devname varchar(100),
        @before_size int,
        @after_size int,
        @before_time datetime,
        @after_time datetime,
        @error int

if (@status&1) = 1
begin
```

```
    print "LOG FULL: database '%1!'", @dbname
end

if @segmentname = (select name from syssegments
            where segment = 2)
begin
    select  @before_time = getdate(),
        @before_size = reserved_pgs(id, doampg)
    from sysindexes
    where sysindexes.name = "syslogs"

    print "LOG DUMP: database '%1!', threshold '%2!'",
        @dbname, @space_left

select @devname = "/backup/" + @dbname + "_" +
        convert(char(8), getdate(),4) + "_" +
        convert(char(8), getdate(), 8)

dump transaction @dbname to @devname
    /* error checking */
    select @error = @@error
    if @error != 0
    begin
        print "LOG DUMP ERROR: %1!", @error
    end

/* get size of log and time after dump */
    select @after_time = getdate(),
        @after_size = reserved_pgs(id, doampg)
        from sysindexes
        where sysindexes.name = "syslogs"

    print "LOG DUMPED TO: device '%1!", @devname
    print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
        @before_size, @after_size
    print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time
end

else

begin /* continued on next page */
    print "THRESHOLD WARNING: database '%1!', segment '%2!' at
'%3!' pages", @dbname, @segmentname, @space_left
end
```

a. What message is printed if the procedure is called as the result of reaching the log's last-chance threshold?

b. What is the segment ID for the transaction log segment?

c. What information is written to the error log if the log segment is the threshold segment?

d. How could the system be set up so that all segment thresholds point to the **sp_thresholdaction** procedure in **sybsystemprocs**?

e. What information is written to the error log if the threshold segment is not the log segment?

f. What would you change in the procedure to dump the transaction to a tape device named "/dev/nrst1"

g. If the log does not shrink much after the dump transaction, what could be happening?

10. You want to override the above **sp_thresholdaction** procedure for your database **dbNN**. The above procedure cannot be changed. You want to set the threshold for the data segment of that database at 200 pages. The threshold procedure then prints an error message to the error log. Answer the following questions:
    a. Could you create a procedure of the same name, **sp_thresholdaction**, in the database? Why or why not?

    b. What procedure would you follow to set this up for your database?

11. You have a table assigned to a specific segment. You want to report when that table reaches 25%, 50%, 65%, and 80% capacity of the segment. This reporting should be done automatically. The total number of available pages in the segment is 1000.
    a. What do you do (list the commands, also)?

b. Write a simple free space threshold procedure named **space_monitor** that prints a message to the error log indicating the number of pages left.

FINDING ROOT CAUSES AND PROPOSING SOLUTIONS

12. A user calls and tells you that he/she has executed a transaction but the system just hangs, not returning control to the client application.
    a. Before looking at the error log, what are two possible reasons this might happen?

b. There are actually conflicting locks on the table this user is attempting to access. How do you find who is causing you to hang and the name of the object in the database that is being locked out?

13. You receive the following error after trying to log into server as sa:

```
% isql -Usa -P -SMARION10
Operating-system error:
        No such file or directory
DB-LIBRARY error:
        Could not open interface file.
%
```

What is the problem? How would you find the problem and then fix it?

AUDITING

14. A particularly sensitive table containing confidential information needs to keep track of user activity and provide the data for reporting this activity. You have attempted to create triggers that would fire and report the information; however, the amount of coding and testing necessary exceeds the time you can allot to create the application's code. You will have to install auditing. How will you plan for use of **sybsecurity**? What steps will you take and what commands will you use to implement this plan? The following information regarding the auditing of the database is defined:
    a. 100 writes per day, 5 days per week on the audit system.
    b. An audit record can vary between a minimum of 20 to a maximum of 420 characters.

c. Audit system is backed up, then purged every three months (13 weeks in a quarter). There is about a one-week overlap from quarter to quarter.

d. Audit system is not mirrored.

e. Audit system is maintained on separate device.

f. You will audit inserts, updates, deletes, and selects on the table in your db.

LOGINS, ROLES, AND PERMISSIONS

15. The following requirements for your new database installation have just been given to you. Write out the appropriate commands, then implement them on the new Server.

    a. User harry42 reports that he has forgotten his password. What steps would you take to provide him with a new one?

b. New employee Marie needs access to the SQL Server. Her login will be'marie42', with 'friday' as her password. Her default database is to be **master**. She will be responsible for database backups and restores, so will require the appropriate role. Outline the steps necessary to enable her to perform her tasks, then verify that the steps were performed.

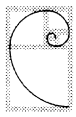
c. Now that Marie has taken over backup responsibilities, tom42 no longer requires the 'OPER' role. How would you make this change?

d. tom42, in turn, will be heading up a new project for which he will need to create and administer databases, including logins. What changes need to be made to his account to enable him to perform these tasks?

Where to Go From Here: Education

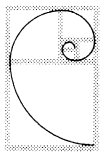
- SQL Server
- Application Programming
- Open Server Programming
- System Administration
- Relational Design
- Performance & Physical Design
- Front End Products

SQL Server	System 10 Introduction to SQL System 10 Fast Track to SQL Server Sybase For End Users
Application and Open Server Programming	Open Client Open Server
System Administration	System 10 System and Database Administration System and Database Administration for Netware (4.2)
Relational Design	Relational Database Design
Performance & Physical Design	Performance & Tuning
Front End Products	APT Workbench Report Workbench Fundamentals of SQR/Advanced SQR/Easy SQR Structured Techniques & DEFT Gain*Momentum* for Managers Gain*Momentum* for Developers

SYBASE 

Where To Go From Here: Documentation

- Sybase documentation related to SQL Server and database administration:
  - SYBASE SQL Server Installation Guide
  - SYBASE SQL Server Reference Manual
  - SYBASE SQL Server Utility Programs
  - System Administration Guide for SYBASE SQL Server
  - SYBASE Troubleshooting Guide
  - What's New in System 10



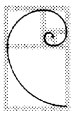
SYBASE®

Module 13

Other Server Administration Topics (Optional)

Objectives

- Use scripts, defncopy, and bulk copy to rebuild a database
- List issues and recommended practices for upgrading
- Configure SQL Server for remote access
- Describe other kinds of distributed systems: two-phase commit, Replication Server, OmniSQL Gateway
- Describe capabilities of SQL Monitor
- Describe strategies for managing multiprocessor Servers

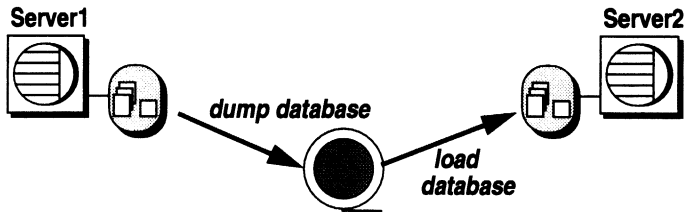
SYBASE 

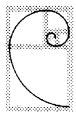
Moving a Database to Another Server

- If database is to be reproduced exactly, on the same kind of machine:
  - `dump database`, `create database`, `load database`
- If not:
  - logical database rebuild (scripts, defncopy)
  - bulk copy

Using dump and load

- Where the database is to be reproduced exactly, use dump and load



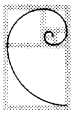
SYBASE 

Logical Database Rebuild

- Where source and target databases are *not* identical, rebuild:
  - Create **scripts**
  - Copy out the objects which have no scripts by using **defncopy** (out)
  - Copy out the data using **bulk copy** (out)
  - Recreate the database, groups, users, permissions, tables using **scripts**
  - Reload the data using **bulk copy** (in)
  - Create indexes
  - Load objects created with **defncopy**
  - Grant permissions
  - Back up the new database

Creating & Organizing Scripts

- Create/update scripts in OS files as you create your databases, objects, users, permissions, etc.
- Use modular structure
  - Group commands by database, in separate files
  - Within a database, group by table or function
  - Start with use database, use if exists/drop, create object, add permissions
  - Separate batches for each command, ending with "go"
  - Remember to use "execute" before stored procedures

SYBASE 

More On Using Scripts

- Use scripts as input files to isql:
  - Unix: `isql -Uabc -Pxyz -i filename`
  - VMS: `isql /u="abc" /p="xyz" /input=filespec`
- Use source control facilities to manage scripts
- If you use a series of scripts frequently, link these in a command file

Unix

Alternative syntax:

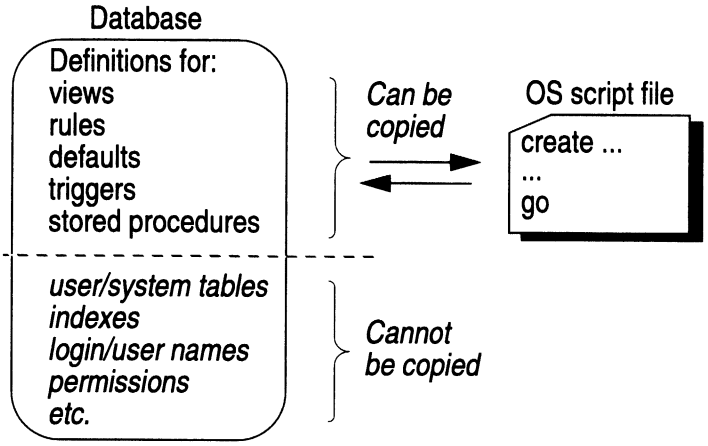
```
isql -Uabc -Pxyz < filename
```

If you use this syntax (redirection) and you do *not* use -P to put the password on the command line, isql will look at the first line of the script file for the password. In that case, be sure the script file is protected.

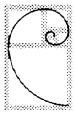
If you use the syntax on the foil ("-i")and do not use -P on the command line, isql will prompt you for the password.

defncopy

- `defncopy` is a stand–alone utility which copies definitions from a database to an OS file or from an OS file to a database



See Utilities manual for your platform.

SYBASE 

defncopy: Basic Syntax

- To copy definition out, creating OS script file:

```
Unix:    defncopy -Ux -Py out
            filename dbname objectname

OpenVMS: defncopy /username=x /password=y out
            filename dbname objectname
```

- To copy definition in, creating object in database

```
Unix:    defncopy -Ux -Py in filename dbname

OpenVMS: defncopy /username=x /password=y
            filename dbname
```

- Examples:

```
defncopy -Urobby -Pplayball out
  deltitle.trig pubs2 deltitle

defncopy /username=robby /password=playball
  in  deltitle.trig  pubs2
```

in | out — specifies direction of copy

filename — name of operating system file destination or source for the definition

defncopy: Notes

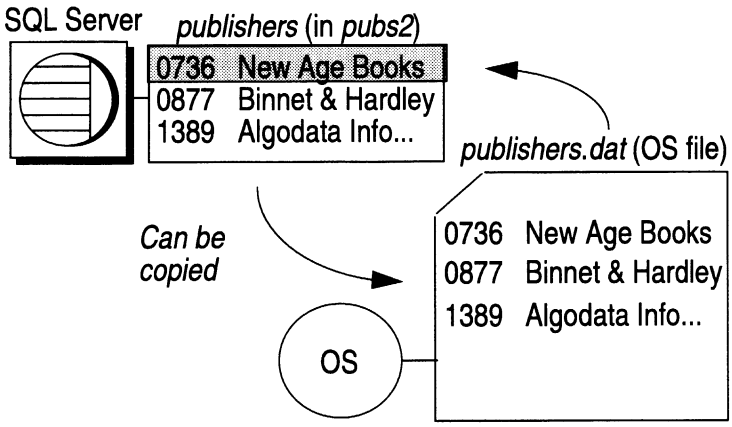
- Permissions and ownership
  - Copying out: User must have *select* permission on object
  - Copying in: User must have permission to create that *type* of object
    User will own the object created
- If you create definition files with an OS editor, end each file with the following comment:

```
/* ### DEFNCOPY:  END OF DEFINITION  */
```

SYBASE

Bulk Copy

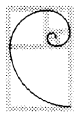
- Bulk copy is a high speed utility for copying table data to or from an Operating System file



Bulk Copy Interfaces

- Three interfaces

bcp:	OS-level stand-alone program
DWB/copy table	visual interface
BCP-Library	Open Client subroutine-level interface

SYBASE 

Bulk Copy: Sample bcp Commands

- bcp
  - Can specify batch size, field terminators, row terminators, and other options on the command line
  - Using -c option: each field copied as a character, separated by tabs, row separated by newline character
- Examples:

```
bcp pubs2..publishers out pub.dat -c -Urobby
bcp pubs2..publishers in pub.dat -c -Urobby
```

- You can supply your password on the command line or be prompted
- You can also save bcp settings in an OS file and use the file as input to bcp

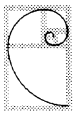
First example Copies out all rows of *pubs2..publishers* to *pub.dat* as character fields. Default field terminator: tab. Default row terminator: new line. (Make sure there are no tabs or new lines in the data or it won't reload.)

Second example Copies data from *pub.dat* to *pubs2..publishers*

Bulk Copy Speed

- Bulk copy *in* is fastest if there are no indexes or triggers on a table
  - "select into/bulkcopy" option must be set for the database with sp_dboption
  - Rules not enforced; defaults *are* enforced
  - Data not logged, but page allocation *is* logged (for recovery purposes)
  - Database checkpointed on completion
- If there are indexes and/or triggers on a table, a slower version of bulk copy is automatically used
  - select into/bulkcopy option does not have to be set
  - Rules not enforced, triggers not fired, defaults *are* enforced
  - Data is logged, as well as page allocation
  - Database checkpointed on completion

SYBASE

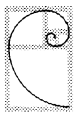


Bulk Copy Suggestions

- When replacing an entire table's data (large)
  - Truncate the table
  - Drop indexes
  - Load data
  - Create indexes
- When adding 10% to 20% or more
  - Drop non–clustered indexes, load data, then create non-clustered indexes
- Otherwise
  - Just load data
- Bulk copying large tables in or out may affect other users' response time

Bulk Copy: Recovery Issues

- Page allocation logged in any case
- What happens if the system fails?
  - If system fails during bulk copy, pages are de-allocated, and no new data remains in database
  - If system fails after bulk copy, all new data remains
- For large tables, set batch sizes so that each batch is a transaction with a checkpoint
  - In that case, recovery is guaranteed up to the last completed batch
  - Typical batch size: 10,000 rows

SYBASE 

Bulk Copy: SA Issues

- For high speed bulk copy in, must set select into/bulk copy to be true for that database

  In master:
  ```
  sp_dboption "salesdb", "select
     into", true
  go
  ```

  Then:
  ```
  use salesdb
  go
  checkpoint
  go
  ```

- After bulk copy in, back up the database

Lab 13a: Logical Database Rebuild

In this lab, we ask you to use scripts, defncopy, and bulk copy to create and load a table similar to the publishers table in pubs2.

1. Write a small "create table" script that creates a table in your database that is structured like *pubs2..publishers*. Test it out, then drop the table you just created.

2. Use `defncopy` to copy out *pub_idrule*. Add the resulting code to your script. Run your new script, and see if the table and rule are created properly.

3. Load your new table with data from *pubs2..publishers* using bulk copy commands shown in this module. Check to see that the data was loaded.

Upgrading

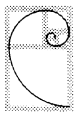
- Follow procedure in Installation Guide!
- Make sure the new version of the software is certified for the OS version
- Have on hand up-to-date scripts, or hard copies of important system tables in master
- If you need to change SQL Server's sort order, do so *after* the upgrade
- Test out new features and functionality in a test environment
- Check if new release has additional reserved words that cannot be used as identifiers
- Remember: You cannot load backups from previous releases

changing sort order

- Chapter 14 of the System Administration Guide discusses changing sort order, languages, and character set.

reserved words

- Use `sp_checkreswords` to scan the database for identifiers (tables, columns, user names, triggers, etc.) having the same name as reserved words.
- These need to be changed or quoted: use `set quoted_identifier on` in your batch or procedure to be able to issue a command like `create table "select".`
- How this works: if set, you can use quote marks around those identifiers that have the misfortune of conflicting with a reserved word.

SYBASE 

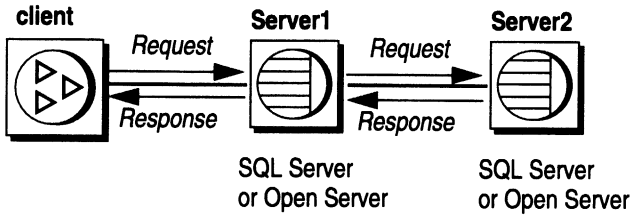
Distributed Database Systems

- Remote procedure calls
- Two-phase commit
- Replication Server
- OmniSQL Gateway

Remote Access

- Remote access is the ability of SQL Server to execute stored procedures on another Server

```
exec Server2.salesdb.dbo.insert_proc
```



- Local Server: the Server the user is connected to
- Remote Server: The "other" Server

Note:

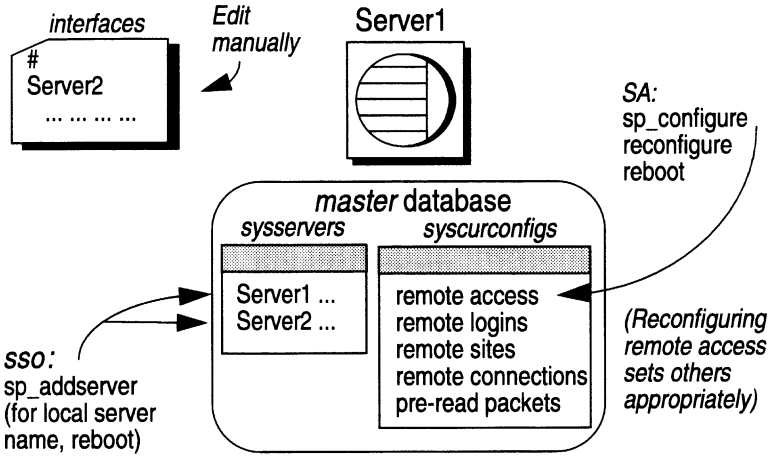
Without remote access, users accessing Server1 could execute procedures only on Server1.

Procedures executed on a remote Server are called "remote procedure calls".

Servers involved may be SQL Servers or Open Servers.

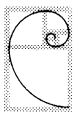
SYBASE

Remote Access Requirements: Local Server (Originating Request)



Steps:

1. Edit interfaces file to add entry for remote Server. (If possible, copy the entry from the *interfaces* file on the remote system.)
2. SSO: use `sp_addserver` to add to *master..syservers* the names of both the local server and the remote server. For the local server name to take effect, you need to reboot.
3. SA: use `sp_configure` to configure for remote access. Then reconfigure, reboot.



SYBASE

Other Server Administration Topics

sp_addserver

- Use sp_addserver to add names of local and remote servers to *master..sysservers*

```
sp_addserver srvname [, {local | null} ] [,
  network_name]
```

- To add the local server name:

```
sp_addserver SERVER1, local
```

  - Reboot for this to take effect

- To add a remote server name, there are two possibilities, depending on which name will appear in *interfaces*:

```
sp_addserver GATEWAY

sp_addserver GATEWAY, null, VIOLET
```

user_defined name to be used in your RPC commands

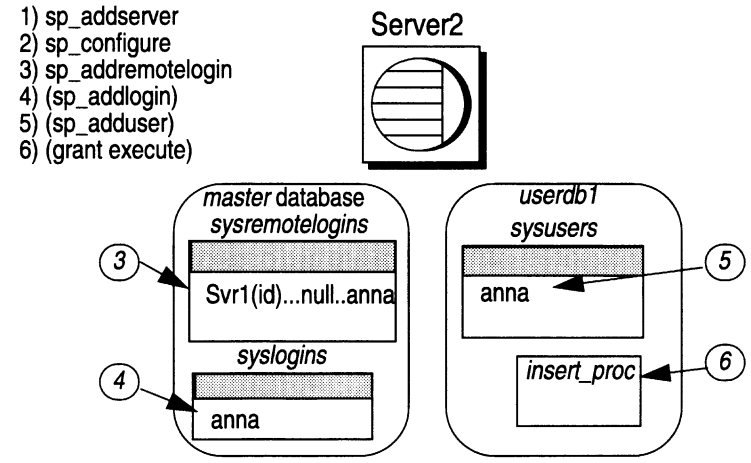
appears in *interfaces* file

sp_addserver GATEWAY In this case, "GATEWAY" will be used in the remote procedure call and is the name in *interfaces*.

sp_addserver GATEWAY, null, VIOLET In this case, the remote procedure call will use "GATEWAY" as the remote server name. The *interfaces* file has the name "VIOLET".

SYBASE

Remote Access Requirements: Remote Server (Receiving Request)



Steps:

1. SSO: use sp_addserver to add to master..syservers the name of the server to be originating request.
2. SA: use sp_configure to configure for remote access. Then reconfigure, reboot.
3. SSO: use sp_addremotelogin to define mappings of remote logins to local logins.
4. If necessary, add appropriate logins.
5. SA or DBO: If necessary, use sp_adduser to add the user to the database on which the stored procedure in question resides.
6. Also, grant execute permission on that stored procedure to that user (unless public has permission).

Mapping Remote Logins

- Use sp_addremotelogin on the "receiving" server

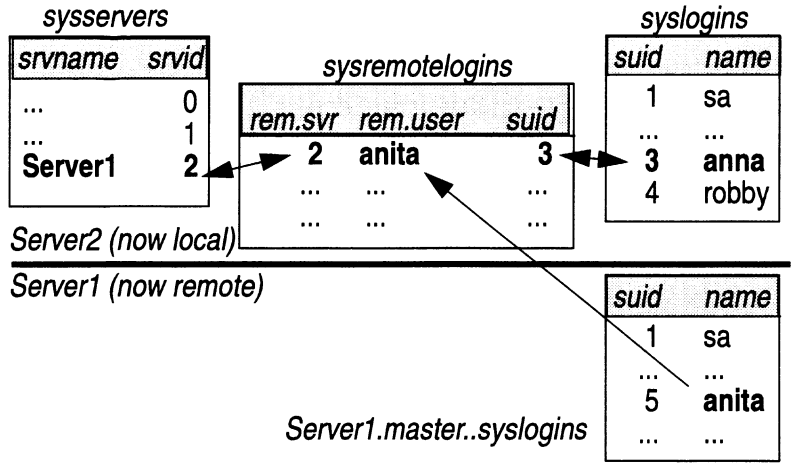
```
sp_addremotelogin remoteserver [, loginame
  [, remotename]]
```

- The *master..sysremotelogins* table associates other servers and their logins with logins on the current server
- Three ways to map remote logins to current server:
  - Map a specific remote login to a specific local login:
  - Map all logins from a certain remote server to one local login:
  - Map all logins from a certain remote server to identical logins on the local server:
- Easiest: map all logins to a specific login name on local server

SYBASE

Mapping Remote Logins (cont.)

```
sp_addremotelogin Server1, anna, anita
```

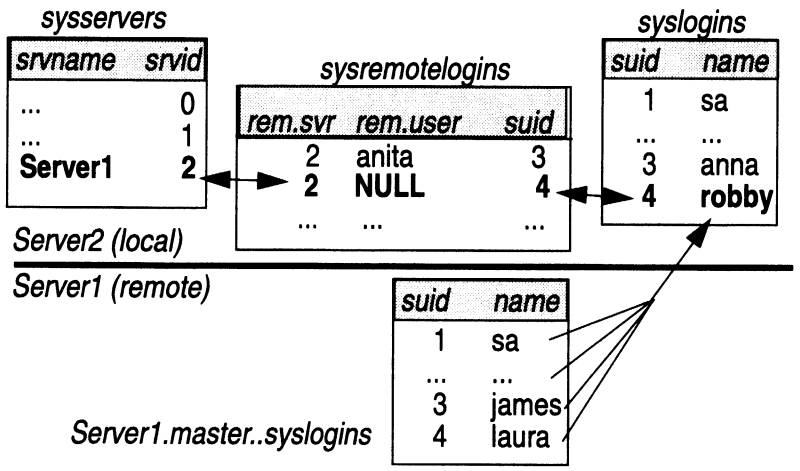


mapping a specific remote login to a specific local login

anita (on *Server1*, *srvid*=2) mapped to *anna (suid*=3).

Mapping Remote Logins (cont.)

```
sp_addremotelogin Server1, robby
```



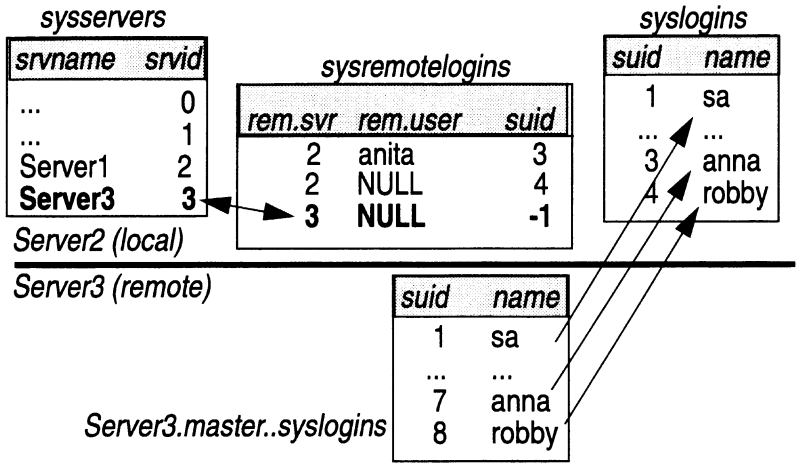
mapping all logins from a certain remote SQL Server to a specific local login

All logins from *Server1 (rvid*=2) mapped to *robby (suid*=4).

SYBASE

Mapping Remote Logins (cont.)

```
sp_addremotelogin Server3
```



mapping all logins from a certain remote SQL Server to identical logins on the local SQL Server

All logins from *Server3 (srvid*=3) are mapped to identical logins on the local SQL Server.

Setting Options

- By default, remote logins are set up "not trusted", i.e., passwords of remote and local login must match
- To reset, use `sp_remoteoption`

```
sp_remoteoption [remote_server, login_name,
  remote_name, opt_name, {true | false}]
```

- Example:

```
sp_remoteoption Server1, anna, anita, trusted,
  true
```

  - Defines remote login *anita*, from Server1, to be trusted (i.e., password not checked with that of login *anna*)
- To reset options related to network handshaking and timing out, use `sp_serveroption`
  - Options: net password encryption, timeouts

trusted/not trusted

Note that if you map all remote logins to a single local login, you must operate in "trusted" mode.

network password encryption

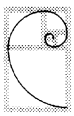
This option specifies whether connections with a remote server are to be initiated with the client side password encryption handshake or the normal (unencrypted password) handshake sequence.

Default: no network encryption, i.e., sent as plain text. (All passwords are encrypted in SQL Server, however, so the encrypted password is what is sent.)

timeouts

When true, disables normal timeout code. Result: site connection handler will not automatically drop the physical connection after a minute with no remote user activity.

Default: false.

SYBASE 

Displaying Information

- To display information about servers and server options, use `sp_helpserver [server]`
  - With an argument, displays information about that server only

```
name     network_name    status                id
-------------------------------------
Server1  Server1   timeouts, net passwd enc 2
```

- To display information about remote logins and options, use `sp_helpremotelogin`
- Syntax:

```
sp_helpremotelogin [remoteserver
    [, remotename]]
```

- With an argument, displays information about that server only

Dropping Remote Servers and Logins

- To drop a server from the list of known servers, use `sp_dropserver`

  ```
  sp_dropserver server [, droplogins]
  ```

  - If you try to drop a server that has remote logins associated with it, the request will be rejected unless you use the droplogins parameter

- To drop remote logins, use `sp_dropremotelogin`

  ```
  sp_dropremotelogin remoteserver [, login_name
    [, remotename] ]
  ```

Lab 13b: Remote Access

Optional (entire lab):

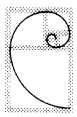
In this lab, we ask you to write the commands you would use to set your SQL Server up for remote access.

1. What steps would you have to go through on your SQL Server to be able to send remote procedure calls to one of the other SQL Servers in the class? Which files or tables would be affected?

Action or (full) command	**Affected file or table**

2. What steps would you have to go through to arrange for another SQL Server to be able to execute procedures on your SQL Server? You would like all attempts from that SQL Server to be mapped to identical logins.

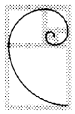
Action or (full) command	**Affected file or table**



Distributed Database Systems

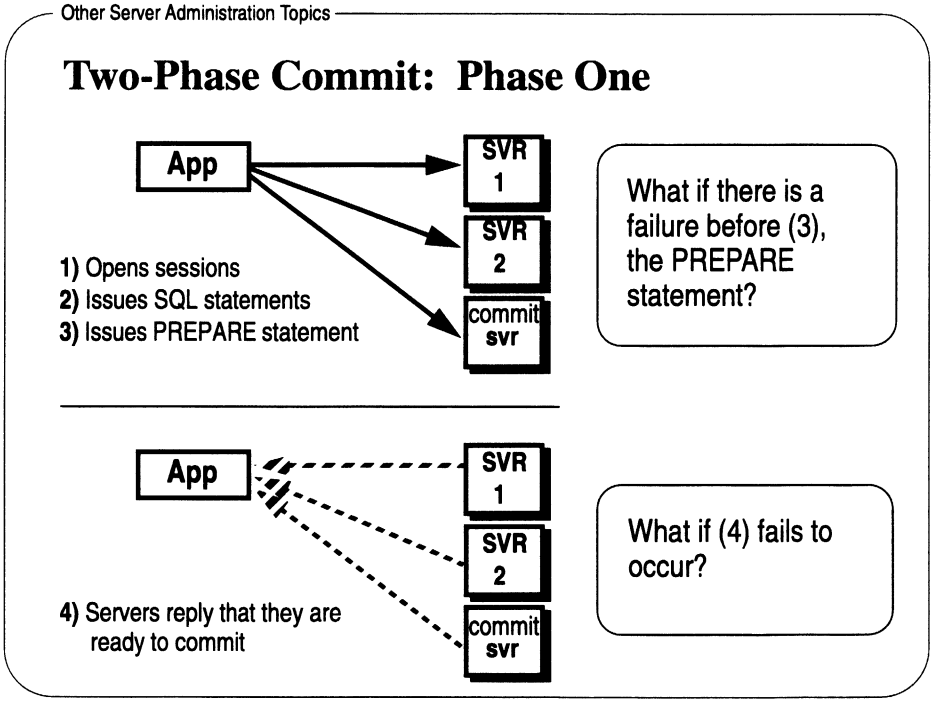
✓Remote procedure calls

- Two-phase commit
- Replication Server
- OmniSQL Gateway

SYBASE 

Two-Phase Commit

- Mechanism to treat separate transactions (which are on separate SQL Servers) as if they were a single transaction
  - Either all transactions involved are committed, or none
- Why is this complicated?
  - Normally recovery of a transaction depends on the log entries for a *single* transaction on a *single* SQL Server
  - In two-phase commit, more than one transaction is involved
  - One SQL Server, the "commit server", is the central record-keeper that helps the application determine whether to commit or roll back
- Application must use Open Client routines to make this happen



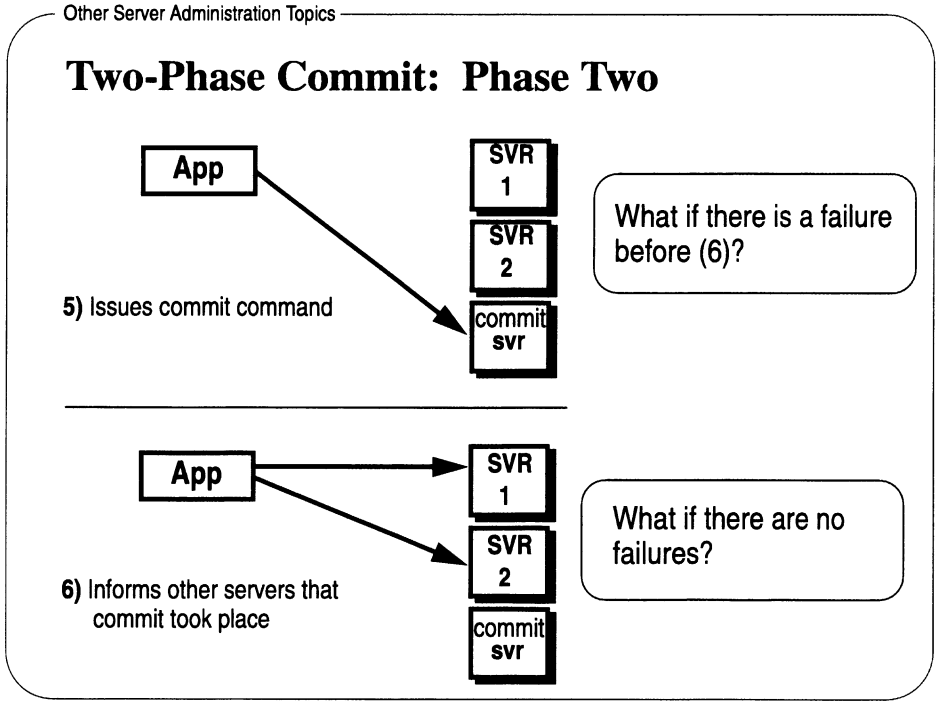
If there is a failure before (3) Transactions are rolled back.

If (4) fails to occur The application will issue an abort statement to all servers, causing a rollback.

commit server There has to be a server that manages the two-phase commit. This server is known as the commit server and may be one of the two servers involved in the two-phase commit.

For details, consult Open Client DB-Library Reference Manual, Section 3.

SYBASE



If there is a failure before (6)...

Using the "probe" process, SQL Servers query commit server to determine whether to mark the transaction to commit or roll back.

If there are no failures...

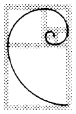
Transaction completes, information about it in *master..spt_committab* on commit server is discarded.

Two-Phase Commit: SA Responsibilities

- Make sure *interfaces* files have entries for commit server, and try to connect to it
- Make sure participating SQL Servers have sufficient user connections
- Monitor the *master..spt_committab* table on the commit server, as it may need to be cleared periodically
- With frequent two-phase commit activity, manage/truncate the log on *master* more often
  - May want to set option to truncate log on checkpoint

user connections

You will need one per server plus one extra one for the commit server.

SYBASE 

Two-Phase Commit: A Distributed Model With Tight Consistency

- Strengths
  - Assures inter-server consistency of data affect by transaction
- Limitations
  - All sites must be online
  - Processing stops when any component fails
  - All tables involved are locked
  - Recovery more complicated
  - Performance depends on slowest part of system

Distributed Database Systems

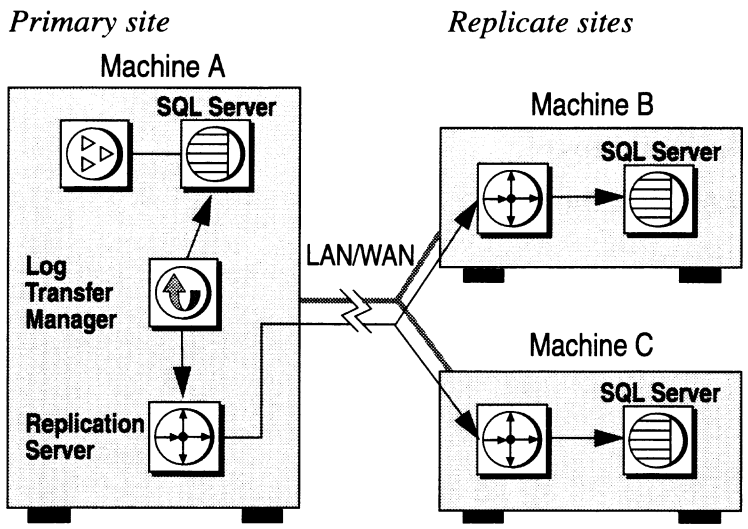
✓Remote procedure calls

✓Two-phase commit

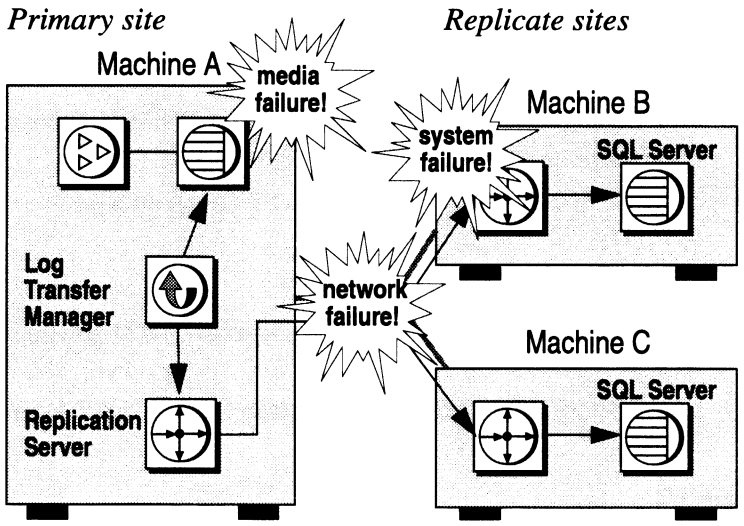
- Replication Server
- OmniSQL Gateway

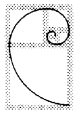
SYBASE

Replication Server



What If There is a System/Network Failure?

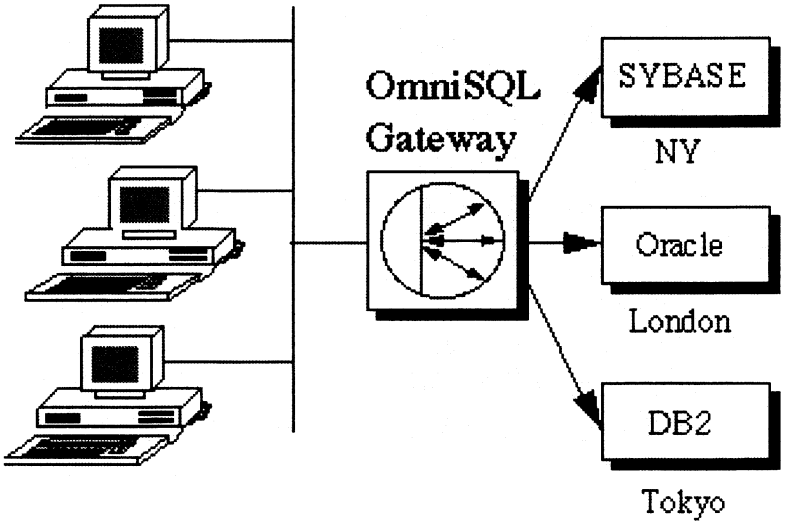


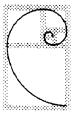
SYBASE 

Replication Server: A Distributed Model With Loose Consistency

- Online asynchronous replication
  - Primary Replication Server forwards committed transactions to other Replication Servers at subscribing sites
  - Processing continues at online sites even if other sites or network go down
- Loose consistency
  - Information not guaranteed identical across sites at any point in time
  - Resynchronization occurs automatically after recovery from failures
- High performance

OmniSQL Gateway



SYBASE 

OmniSQL Gateway (cont.)

- Solves these data interoperabililty problems:
  - What data is available?
  - Where is data located?
  - Which SQL dialect must I use?
  - How can I combine data from multiple sources?
- Offers
  - Heterogeneous data integration
  - Full SQL syntax translation
  - Heterogeneous distributed join
  - Global Transact-SQL stored procedures for Non-SYBASE data

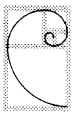
System Administration Tools

Two stand-alone tools:

- **SA Companion**: to manage SQL Server
- **SQL Monitor**: to monitor SQL Server Performance

SA Companion

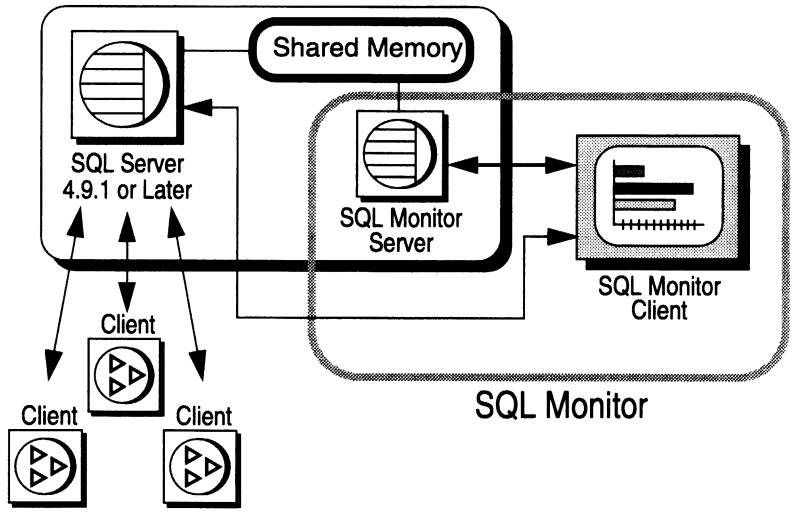
See the SA Companion module for a detailed discussion of this tool. This section will examine SQL Monitor.

SYBASE 

SQL Monitor

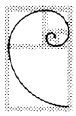
- Provides you with a mechanism to observe SQL Server while it is running
- Accesses SQL Monitor Server and is attached to a single SQL Server (but could be monitoring multiple engines)
- Can help you improve SQL Server performance and efficiency

SQL Monitor (cont.)



SQL Monitor Server — Open Server application that accesses SQL Server shared memory to capture SQL Server performance data

SQL Monitor Client — Client application that connects to SQL Monitor Server and SQL Server and displays performance data in a graphical user interface.

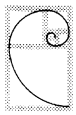
SYBASE 

Questions SQL Monitor Can Help Answer

- Is the configuration optimal?
- Are objects distributed for best performance?
- Where is the processing burden? Where are the bottlenecks?
- Where are the application resource contentions?
- Is the size of my procedure cache appropriate?

What SQL Monitor Can Monitor

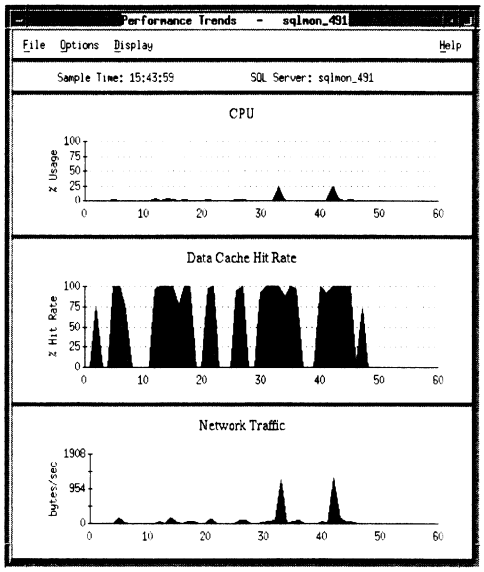
- Memory allocation
- Network traffic
- CPU use
- User process information
- Data and procedure cache use
- Disk I/O volume and average completion time by device
- Transaction activity

SYBASE 

SQL Monitor Features

- Tracks a wide variety of SQL Server performance statistics
- Reads statistics from shared memory rather than querying SQL Server directly
  - Performance impact is minimal
- Presents a graphical, point-and-click interface
  - Can specify which statistics to monitor, the level of detail, and how the information is to be presented
  - Can display data snapshots or continuous data feed with programmable thresholds and sampling rate
  - Can display multiple windows
  - Can choose format, i.e., bar graph vs. line chart, etc.

Monitoring Performance Trends



performance trends window

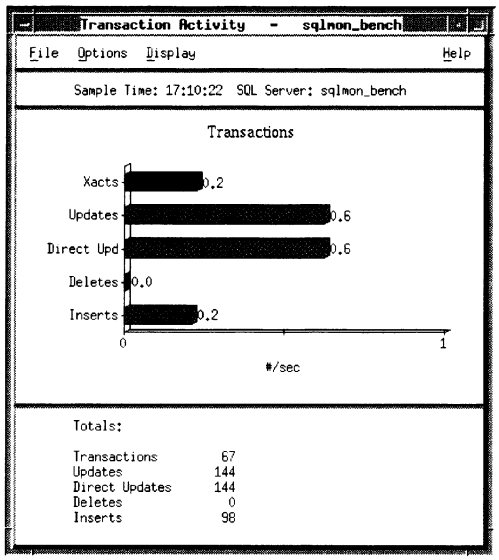
- shows changes in primary SQL Server performance indicators over time
- starts when user opens window; adds data points to the graphs at each sample interval

available indicators

- CPU utilization percentage
- Data cache hit rate
- Device I/O
- Device I/O hit rate
- Network traffic
- Procedure cache hit rate

SYBASE

Monitoring Transaction Activity



transaction activity window Summarizes Transact-SQL transactions executed by SQL Server.

why is this useful? Gives a rough measure of how well SQL Server is performing: a high number of transactions per second generally indicates a high performance level.

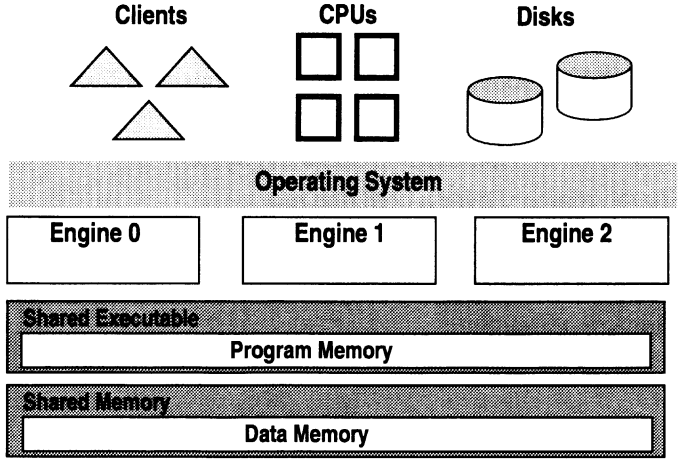
Managing Multiprocessor Servers

- SQL Server implements a Virtual Server Architecture
  - It can take advantage of the parallel processing feature of symmetric multiprocessing (SMP) systems
- SYBASE SMP product is intended for machines with the following features:
  - Symmetric multiprocessing operating system
  - Shared memory over a common bus
  - 1-32 processors
  - No master processor
  - Very high throughput

SYBASE

SQL Server Architecture for SMP

- SQL Server consists of one or more cooperating processes (called engines), all running the server program in parallel



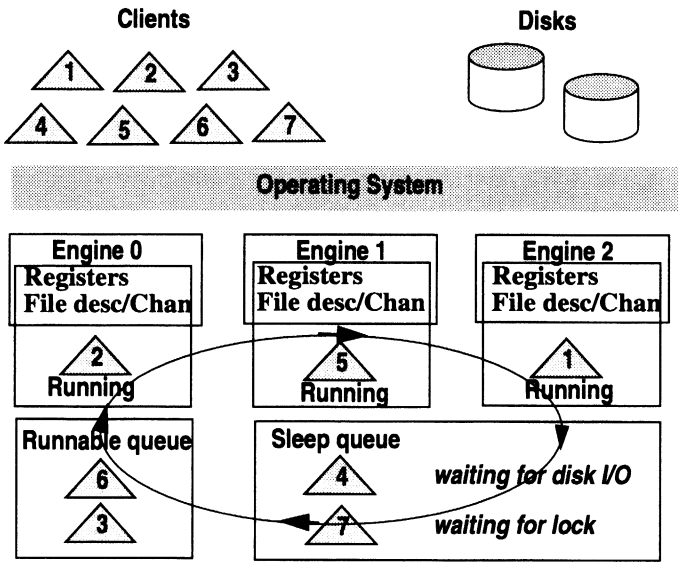
all engines are peers

- any engine can handle any task and use any CPU
- exception: *engine0* handles network management
- engines communicate via shared memory

SYBASE schedules client tasks onto engines

- OS schedules engine processes onto physical processors
- no "affinity" between engines and CPUs—any available CPU is used for any engine

SQL Server Task Management for SMP



1 A client application issues a login request. SQL Server creates a user task to handle work from the client.

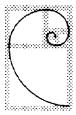
2 The client presents SQL Server with work to do, for example a series of Transact-SQL commands.

3 SQL Server adds the client's user task to the runnable task queue. The server engines compete for the user task at the head of the task queue.

4 The server engine that takes the user task from the queue converts the Transact-SQL commands into low level steps, such as disk I/O.

5 The engine executes each step until the task completes or blocks while waiting for I/O or locking. When the task blocks, it yields the server engine to run other user tasks. Once the block is resolved (i.e., disk I/O completes or a lock is acquired), the user task is again added to the runnable task queue.

6 After the task blocks for the last time, it continues executing until finished. At that time the user task yields the server engine and moves to the sleeping task queue until the client presents the server with more work.

SYBASE 

Strategies for Setting Up Multiple Engines

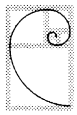
- For optimal performance, you must maintain the right number of engines
  - This is configurable
- When you create a new master device, the system is configured for a single engine
  - To use multiple engines, reset the number of engines

```
sp_configure "max online engines", 3
go
reconfigure
go
```

Then restart SQL Server

Choosing the Right Number of Engines

- Never have more engines than CPUs
  - If a CPU goes offline, reduce "max online engines"
- Have only as many engines as *usable* CPUs
  - One engine per CPU may be excessive if the operating system or other non-server processes take up part of one of the CPUs
- Have enough engines
  - Start with a few engines and add additional ones when the existing CPUs are almost fully used
- Monitor CPU usage with an operating system utility

SYBASE 

Special Considerations for SMP

- Memory
  - SMP may require more memory
  - Increase if error message 701, "There is insufficient memory to run this query", appears frequently
  - Do not use `dbcc memusage`—processes may be lost
- Disks
  - SMP may increase relative demands on disks
  - Important to spread heavily-used data across multiple disks

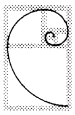
dbcc memusage

`dbcc memusage` is an undocumented command and is not officially part of the product. It can be a useful tool, however, particularly if you do not have access to SQL Monitor.

Avoid using `dbcc memusage` with multiple engines, as it can result in the loss of processes.

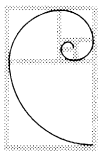
Special Considerations for SMP (cont.)

- Application Design
  - Increased chance of contention as multiple processes try to access same page
  - Important to follow principles of good database design:
    - No more than 2 or 3 indexes on heavily-updated table
    - Low fillfactor in `create index` commands
    - Short transactions
    - Use temporary tables (in *tempdb*) when possible

SYBASE 

Summary

- Scripts, defncopy, and bulk copy can be used to rebuild a database
- Distibuted systems:
  - remote procedure calls
  - two-phase commit
  - Replication Server
  - OmniSQL Gateway
- Administration/Monitoring tools: SA Companion, SQL Monitor
- Running SQL Server in an SMP environment
  - Very high throughput
  - Important to maintain right number of engines



SYBASE®

Appendix A

Suggested Readings

Recommended SYBASE SQL Server and RDBMS Titles

Client/Server Architecture
Alex Berson
McGraw-Hill, 1992.
ISBN 0-07-005076-7

Database Tuning: A Principled Approach
Dennis E. Shasha
Prentice Hall, 1992

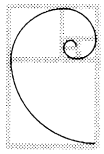
The Guide to SQL Server
Aloke Nath
Addison-Wesley, 1990. Second printing, July 1991
ISBN 0-201-52336-1

A Guide to the SQL Standard: A User's Guide
C.J. Date with Hugh Darwen
Addison-Wesley, 1993
ISBN 0-201-55822-X

A Guide to Sybase and SQL Server
D. McGoveran with C.J. Date
Addison-Wesley, 1992
ISBN 0-201-55710-X

An Introduction to Database Systems, Vol. I, 5th edition
C.J. Date.
Addison-Wesley, 1990. Reprinted with corrections, February 1991.
ISBN 0-201-51381-1

An Introduction to Database Systems, Vol. II
C.J. Date
Addison-Wesley, 1983. Reprinted with corrections, July 1984.
ISBN 0-201-14474-3



SYBASE®

Appendix B

Useful Procedures

Useful Procedures

Disclaimer: This procedures is not supported by Sybase.

sp_helpdisk

Alphabetically by database, show all database device fragmentsand size in MB. Then, alphabetically by device, show all database device fragments and size in MB. Then, alphabetically by device, show all device names, physical device names, size and vdevno.

```
use master
go
drop proc sp_disk
go
create proc sp_disk
as
set nocount on
select
   logical_name = substring(name, 1, 12),
   physical_name = substring(phyname, 1, 25),
   vdevno = low / power(2, 24),
   size = (high - low + 1) * 2 / 1024
from
   master.dbo.sysdevices
where
   status & 2 = 2
order by
   vdevno
go
grant execute on sp_disk to public
go
```

Disclaimer: This procedures is not supported by Sybase.

sp_freedisk

Alphabetically by device, show physical name, vdevno, size in MB, reserved space in MB, space left in MB.

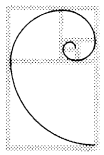
```
use master
go
drop proc sp_freedisk
go
create procedure sp_freedisk
as
set nocount on
select
   logical_name = substring(d.name, 1, 12),
   physical_name = substring(d.phyname, 1, 17),
   vdevno = d.low / power(2, 24),
   size = (d.high - d.low + 1) * 2 / 1024,
   reserved = isnull(sum(u.size) * 2 / 1024, 0),
   left = (d.high - d.low + 1) * 2 / 1024 -
      isnull(sum(u.size) * 2 / 1024, 0)
from
   master.dbo.sysdevices d, master.dbo.sysusages u
where
   d.status & 2 = 2
   and u.vstart / power(2, 24)=*d.low / power(2, 24)
group by
   substring(d.name, 1, 12),
   substring(d.phyname, 1, 17),
   d.low / power(2, 24),
   (d.high - d.low + 1) * 2 / 1024
order by
   vdevno
go
grant execute on sp_freedisk to public
go
```

Disclaimer: This procedures is not supported by Sybase.

sp_disk

Sorted by vdevno, show all devices, their physical name and size.

```
use master
go
drop proc sp_disk
go
create proc sp_disk
as
set nocount on
select
   logical_name = substring(name, 1, 12),
   physical_name = substring(phyname, 1, 25),
   vdevno = low / power(2, 24),
   size = (high - low + 1) * 2 / 1024
from
   master.dbo.sysdevices
where
   status & 2 = 2
order by
   vdevno
go
grant execute on sp_disk to public
go
```

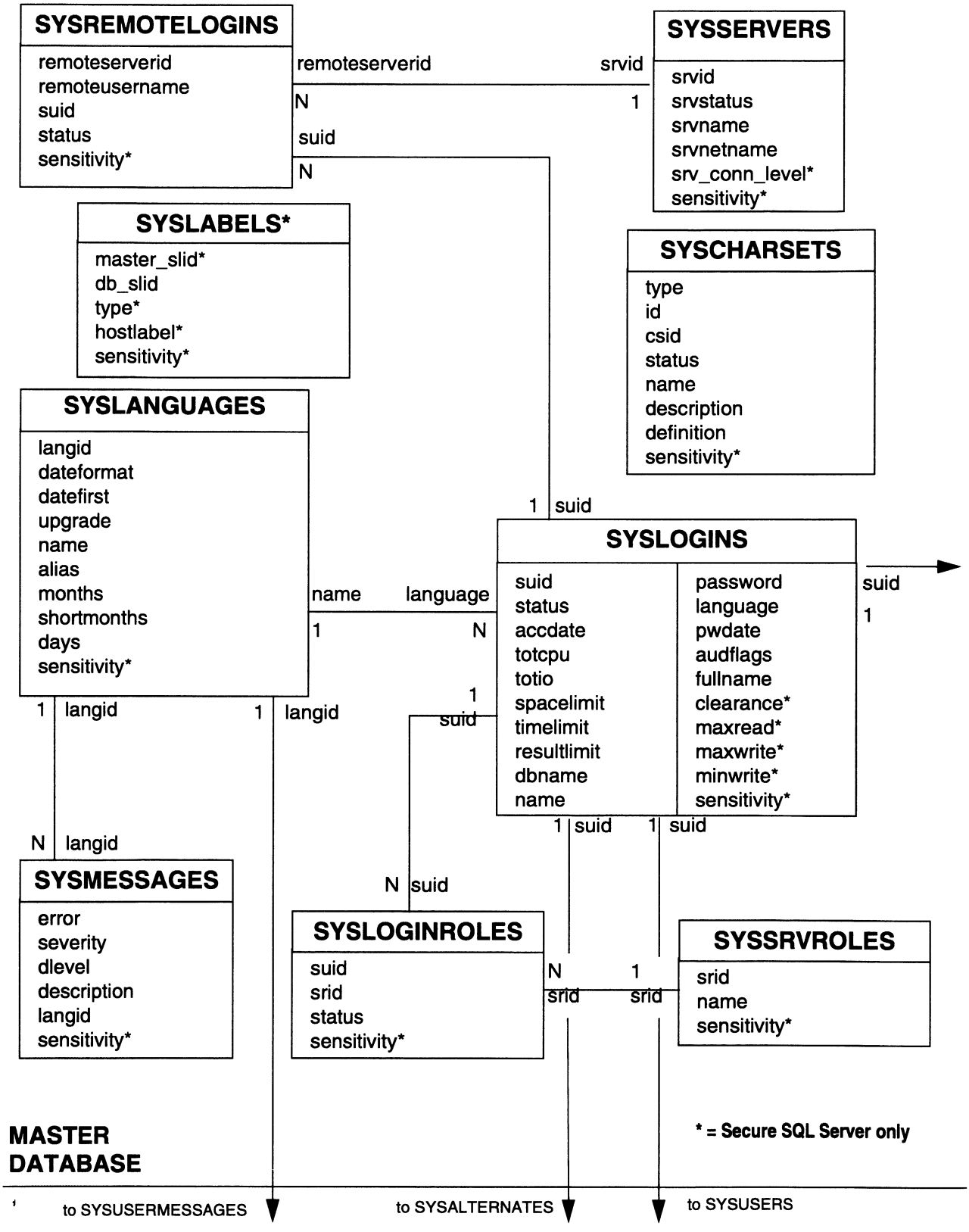


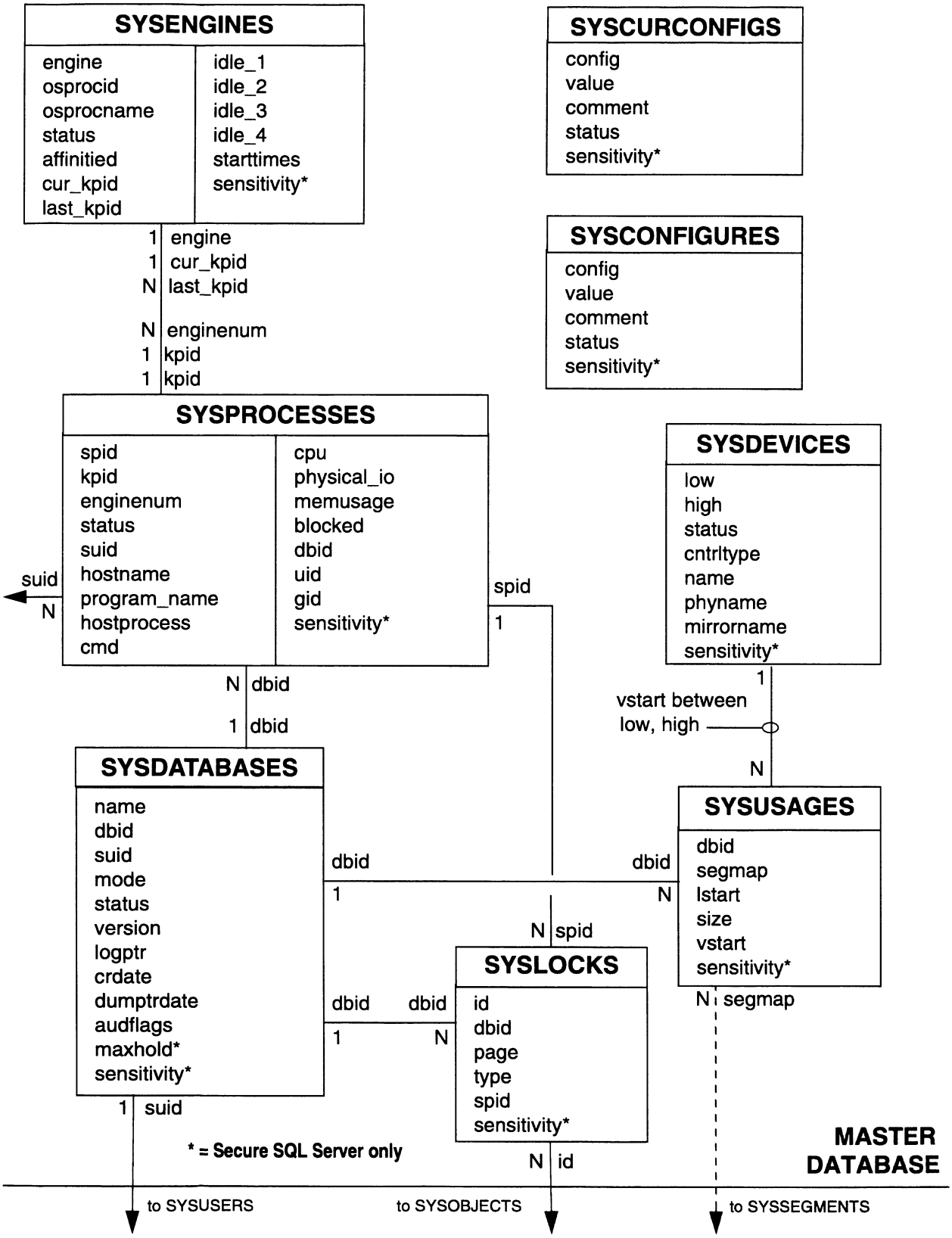
SYBASE®

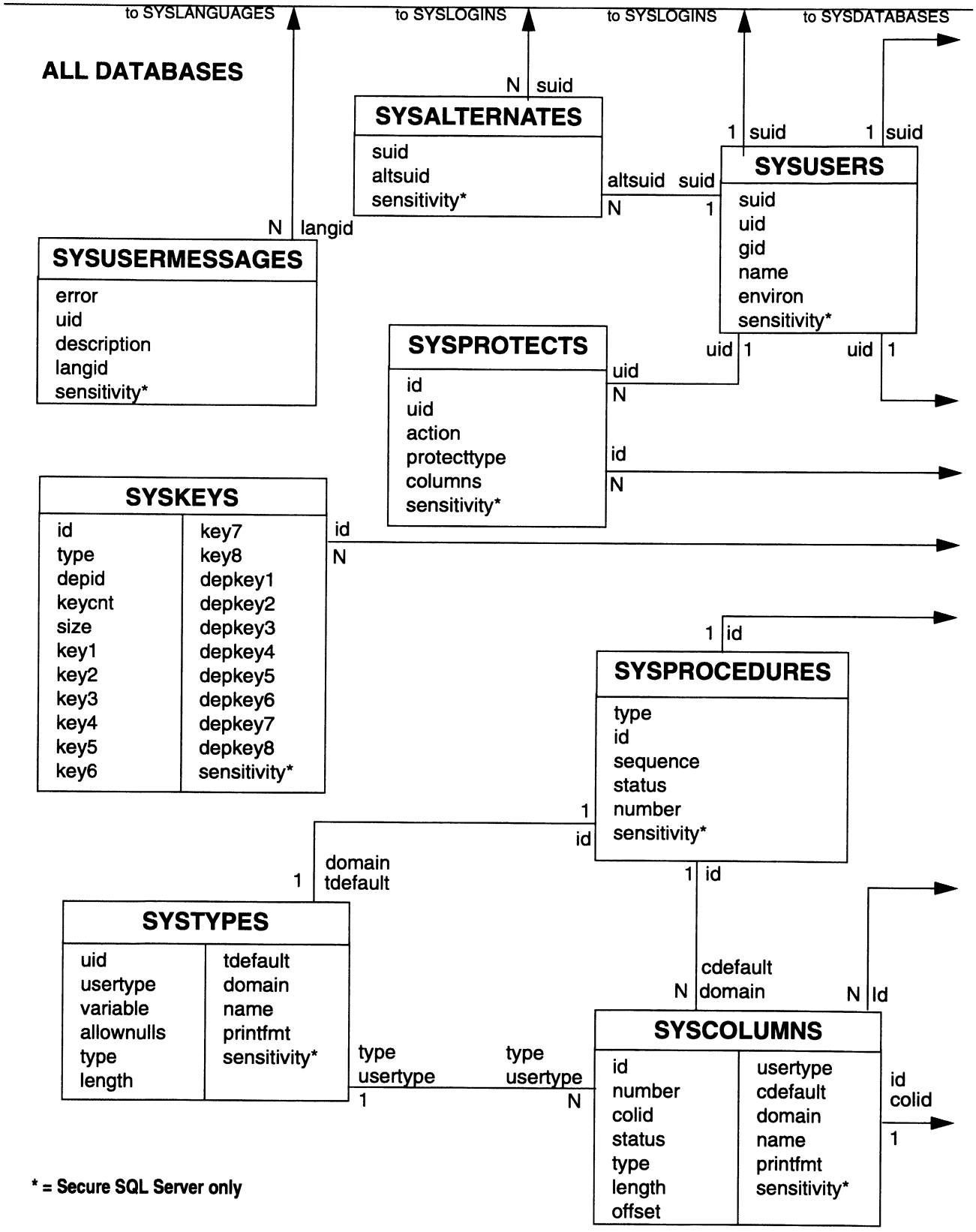
Appendix C

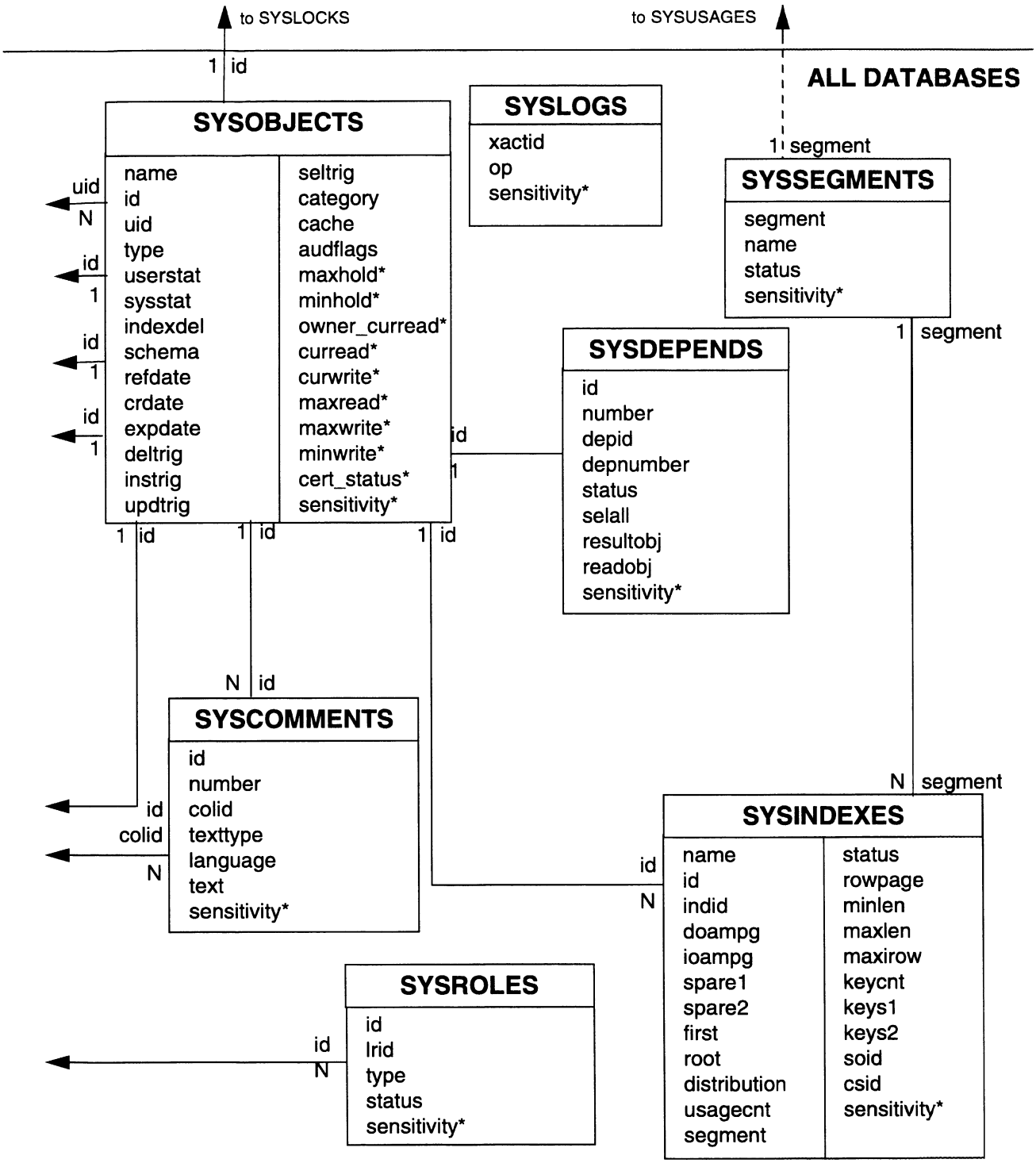
System Tables

System Tables

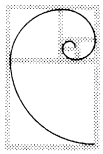



SYSENGINES
engine
osprocid
osprocname
status
affinitied
cur_kpid
last_kpid
idle_1
idle_2
idle_3
idle_4
starttimes
sensitivity*
SYSCURCONFIGS
config
value
comment
status
sensitivity*
SYSCONFIGURES
config
value
comment
status
sensitivity*
1 engine
1 cur_kpid
N last_kpid
N enginenum
1 kpid
1 kpid
SYSPROCESSES
spid
kpid
enginenum
status
suid
hostname
program_name
hostprocess
cmd
cpu
physical_io
memusage
blocked
dbid
uid
gid
sensitivity*
suid
N
spid
1
SYSDEVICES
low
high
status
cntrltype
name
phyname
mirrorname
sensitivity*
1
vstart between
low, high
N
N dbid
1 dbid
SYSDATABASES
name
dbid
suid
mode
status
version
logptr
crdate
dumptrdate
audflags
maxhold*
sensitivity*
dbid
1
dbid
N
SYSUSAGES
dbid
segmap
lstart
size
vstart
sensitivity*
N spid
SYSLOCKS
id
dbid
page
type
spid
sensitivity*
dbid
1
dbid
N
N segmap
1 suid
* = Secure SQL Server only
N id
MASTER DATABASE
to SYSUSERS
to SYSOBJECTS
to SYSSEGMENTS


to SYSLANGUAGES
to SYSLOGINS
to SYSLOGINS
to SYSDATABASES
ALL DATABASES
N suid
SYSALTERNATES
suid
altsuid
sensitivity*
1 suid
1 suid
SYSUSERS
suid
uid
gid
name
environ
sensitivity*
altsuid suid
N 1
N langid
SYSUSERMESSAGES
error
uid
description
langid
sensitivity*
SYSPROTECTS
id
uid
action
protecttype
columns
sensitivity*
uid 1
uid 1
uid
N
id
N
SYSKEYS
id
type
depid
keycnt
size
key1
key2
key3
key4
key5
key6
key7
key8
depkey1
depkey2
depkey3
depkey4
depkey5
depkey6
depkey7
depkey8
sensitivity*
id
N
1 id
SYSPROCEDURES
type
id
sequence
status
number
sensitivity*
1
id
1 id
1
domain
tdefault
SYSTYPES
uid
usertype
variable
allownulls
type
length
tdefault
domain
name
printfmt
sensitivity*
cdefault
N domain
N Id
SYSCOLUMNS
id
number
colid
status
type
length
offset
usertype
cdefault
domain
name
printfmt
sensitivity*
type
usertype
1
type
usertype
N
id
colid
1
* = Secure SQL Server only



*** = Secure SQL Server only**



SYBASE®

Appendix D

Lab Answers

Lab 1 – Overview

1. List 3 features of Client/Server architecture:

Hardware interoperability
Software interoperability
The ability to make optimal use of clients and data servers

2. List 5 administrative responsibilities:

Start and stop SQL Server
Allocate resources
Create, drop, and enlarge databases
Create and manage SQL Server login accounts
Configure SQL Server
Maintain regular backups, and restore as needed
Monitor, tune, and troubleshoot SQL Server

3. Log into SQL Server using isql. (Ask instructor for instructions on username and password, and fill these in on the information sheet on the next page.) Run sp_helpdb to display the databases on your server.

```
godzilla% isql -Ustudent42 -Pstudent42
1> sp_helpdb
2> go
name                 db_size      owner                 dbid
        created
        status

 ---------------------- ------------ ----------------------
------
       --------------

-------------------------------------------------------------------
-------
------------------------------
 master                        3.0 MB sa                      1
        Jan 01, 1900
        no options set

 model                         2.0 MB sa                      3
        Jan 01, 1900
        no options set

 pubs2                         2.0 MB sa                      6
        Nov 01, 1993
        no options set

 sybsecurity                    5.0 MB sa                     5
        Nov 01, 1993
        trunc log on chkpt
```

```
sybsystemprocs                  10.0 MB sa                              4
        Nov 01, 1993
        trunc log on chkpt

tempdb                          2.0 MB sa                               2
        Nov 01, 1993
        select into/bulkcopy


(return status = 0)
```

4. Exit from isql. Write a short script which creates a table in tempdb called testN, where N is your user number. Run the script using isql. Log into SQL Server again, use tempdb, and display the table structure using sp_help testN. (Hint: Your script will have two batches: one with use tempdb, one which creates the table.)

```
/* createtable script */

use tempdb
go
create table test42(a int, b char(10))
go
```

Running it:

```
godzilla% isql -Ustudent42 -Pstudent42 < createtable.script
```

Checking results:

```
godzilla% isql -Ustudent42 -Pstudent42
1> use tempdb
2> go
1> sp_help test42
2> go
 Name                            Owner
        Type
 ------------------------------ ------------------------------
       ----------------------
 test42                          dbo
        user table

 Data_located_on_segment        When_created
 ------------------------------ --------------------------
 default                               Nov  1 1993  3:28PM

 Column_name     Type            Length Prec Scale Nulls
Default_name
        Rule_name       Identity
 --------------- --------------- ------ ---- ----- -----
---------------
        --------------- --------
 a               int                  4 NULL  NULL     0 NULL
        NULL                    0
 b               char                10 NULL  NULL     0 NULL
        NULL                    0
```

```
Object does not have any indexes.
No defined keys for this object.

(return status = 0)
```

5. [Optional] Consult the entity relationship diagram of system tables in the Appendix to this course and write select statements to:

(a) display the text of error message 1205

```
1> use master
2> go
1> select description from sysmessages
2> where error = 1205
3> go

         description

------------------------------------------------------------------------
-------
------------------------------------------------------------------------
---------------
------------------------------------------------------------------------
---------------
-----------------------

         Your server command (process id #%d) was deadlocked with
another process and has been chosen as deadlock victim.  Re-run
your command.


(1 row affected)
```

(b) using a join between two tables, list the names of the databases on your SQL Server alongside the login names of their owners

```
1> select sysdatabases.name dbname, syslogins.name owner
2> from sysdatabases, syslogins
3> where sysdatabases.suid = syslogins.suid
4> go
 dbname                         owner
 ------------------------------ ------------------------------
 master                         sa
 model                          sa
 pubs2                          sa
 sybsecurity                    sa
 sybsystemprocs                 sa
 tempdb                         sa

(6 rows affected)
```

Lab 2 - Installation

1. If you installed a SQL Server called VIOLET, what files would be created or modified?

Created:

interfaces (modified if already exists)

RUN_VIOLET, RUN_SYB_BACKUP (or RUN_BackupServerName)

errorlog_VIOLET (created at startup)

backup.log

master device

2. Look in the *sybase* directory tree to find which SQL Server or Servers are known to your system.

a) Which SQL Server(s) are installed on your machine?

Check interfaces file for entries with your host name

or

List the RUNSERVER files in the install directory

b.) Which are running?

Use showserver. For example:

```
godzilla% showserver
USER       PID %CPU %MEM    SZ   RSS TT STAT START   TIME COMMAND
sybase   19359  0.0 13.8   700 4168 ?  S    11:57    3:05
/sybase/server10/bin/dataserver
-d/sybase/server10/devices/KINGKONG10_master.dat -sKINGKONG10
-e/sybase/server10/install/KINGKONG10_errorlog -i/sybase/server10
-M/sybase/server10/install
sybase   19264  0.0  2.8   700  852 ?  S    11:46    3:08
/sybase/server10/bin/dataserver
-d/sybase/server10/devices/GODZILLA10_master.dat -sGODZILLA10
-e/sybase/server10/install/GODZILLA10_errorlog -i/sybase/server10
-M/sybase/server10/install
sybase   19362  0.0  0.0 3560    0 ?  IW   11:59    0:00
/sybase/server10/bin/backupserver -SKINGKONG10_BACKUP
-e/sybase/server10/install/KINGKONG10_backup.log
-I/sybase/server10/interfaces -M/sybase/server10/bin/sybmultbuf
sybase   19285  0.0  0.0 3560    0 ?  IW   11:47    0:00
/sybase/server10/bin/backupserver -SGODZILLA10_BACKUP
-e/sybase/server10/install/GODZILLA10_backup.log
-I/sybase/server10/interfaces -M/sybase/server10/bin/sybmultbuf
```

Here there are two SQL Servers running, KINGKONG10 and GODZILLA10 (see dataserver command line). There are also two backup servers running: KINGKONG10_BACKUP and GODZILLA10_BACKUP (see backupserver command line).

3. For a SQL Server that is running, look through the files online and recreate a diagram similar to the one at the beginning of this module. Include the names of relevant files and their content (as it pertains to the SQL Server in question).

(Answer depends on installation on student machine.)

4. Answer the following using the RUNSERVER and interfaces files:

a) Where is the master device, and how big is it?

This example uses the KINGKONG10 server. First, here's the runserver file:

```
godzilla% cat RUN_KINGKONG10
#!/bin/sh
#
# SQL Server Information:
# name:                         KINGKONG10
# master device: /sybase/server10/devices/KINGKONG10_master.dat
# master device size:           8704
# errorlog:       /sybase/server10/install/KINGKONG10_errorlog
# interfaces:                   /sybase/server10
# shared memory file location:  /sybase/server10/install
#
/sybase/server10/bin/dataserver \
-d/sybase/server10/devices/KINGKONG10_master.dat -sKINGKONG10 \
-e/sybase/server10/install/KINGKONG10_errorlog \
-i/sybase/server10 \
-M/sybase/server10/install
```

The master device is in /sybase/server10/devices/KINGKONG10_master.dat, and is 8704 pages, or 17 Mb in size. To confirm:

```
godzilla% cd /sybase/server10/devices
godzilla% ls -l KINGKONG10_master.dat
-rw-rw-r--  1 sybase   17825792 Nov  1 15:38 KINGKONG10_master.dat
```

b) What ports is the Server using?

From the interfaces file:

```
## KINGKONG10 on godzilla using tcp
##       Services :   query (1655)  master (1655)
KINGKONG10
        query tcp ether godzilla 1655
        master tcp ether godzilla 1655
```

This shows the KINGKONG10 server to be using port 1655.

c) Where is the errorlog?

From the RUN_KINGKONG10 file:

```
/sybase/server10/bin/dataserver \
-d/sybase/server10/devices/KINGKONG10_master.dat -sKINGKONG10 \
-e/sybase/server10/install/KINGKONG10_errorlog \
-i/sybase/server10 \
-M/sybase/server10/install
```

The -e flag shows the errorlog is in /sybase/server10/install/errorlog_KINGKONG10.

d) Which of these can you be certain is accurate?

The location of the master device, and of the errorlog as stated in the dataserver command line will be correct. The size of the master device, and any other information included in the comment lines of the runserver file reflect the state of the server at installation, and are not automatically updated to reflect changes performed later. Thus, they cannot necessarily be relied upon. **(Do update these comments if you make changes!)**

5. Run `showserver`. What information does `showserver` give you?

```
godzilla% cd /sybase/server10/install
godzilla% showserver
USER       PID %CPU %MEM   SZ  RSS TT STAT START  TIME COMMAND
sybase   19359  0.0 13.8  700 4168 ?  S    11:57   3:05
/sybase/server10/bin/dataserver
-d/sybase/server10/devices/KINGKONG10_master.dat -sKINGKONG10
-e/sybase/server10/install/KINGKONG10_errorlog -i/sybase/server10
-M/sybase/server10/install
sybase   19264  0.0  2.8  700  852 ?  S    11:46   3:08
/sybase/server10/bin/dataserver
-d/sybase/server10/devices/GODZILLA10_master.dat -sGODZILLA10
-e/sybase/server10/install/GODZILLA10_errorlog -i/sybase/server10
-M/sybase/server10/install
sybase   19362  0.0  0.0 3560    0 ?  IW   11:59   0:00
/sybase/server10/bin/backupserver -SKINGKONG10_BACKUP
-e/sybase/server10/install/KINGKONG10_backup.log
-I/sybase/server10/interfaces -M/sybase/server10/bin/sybmultbuf
sybase   19285  0.0  0.0 3560    0 ?  IW   11:47   0:00
/sybase/server10/bin/backupserver -SGODZILLA10_BACKUP
-e/sybase/server10/install/GODZILLA10_backup.log
-I/sybase/server10/interfaces -M/sybase/server10/bin/sybmultbuf
```

It shows, among other things, that there are 2 SQL Servers running on this machine: GODZILLA10 and KINGKONG10, both of which are using Unix files for their master device. It also shows the location of each Server's errorlog, shared memory files and the interfaces file being used.

There are also two backup servers running: GODZILLA10_BACKUP and KINGKONG10_BACKUP.

6. Scroll through the error log.

```
godzilla% cd /sybase/server10/install
godzilla% cat KINGKONG10_errorlog | more
.
.
.
00:93/11/01 11:58:00.18 kernel  SQL Server/10.0/P/Sun4/OS
4.1.x/1/OPT/Wed Sep 22 12:52:03 PDT 1993
00:93/11/01 11:58:00.18 kernel  Confidential Property of Sybase,
Inc.
00:93/11/01 11:58:00.18 kernel  Copyright Sybase, Inc. 1987, 1993.
00:93/11/01 11:58:00.18 kernel  All rights reserved.
00:93/11/01 11:58:00.18 kernel  Use, duplication, or disclosure by
the United States Government
00:93/11/01 11:58:00.18 kernel  is subject to restrictions as set
forth in FARSsubparagraphs
00:93/11/01 11:58:00.18 kernel  52.227-19(a)-(d) for civilian
agency contracts and DFARS
00:93/11/01 11:58:00.18 kernel  252.227-7013(c)(1)(ii) for
Department of Defense contracts.
00:93/11/01 11:58:00.18 kernel  Sybase reserves all unpublished
rights under the copyright
00:93/11/01 11:58:00.18 kernel  laws of the United States.
00:93/11/01 11:58:00.18 kernel  Sybase, Inc. 6475 Christie Avenue,
Emeryville, CA 94608, USA.
00:93/11/01 11:58:00.19 kernel  Logging SQL Server messages in
file '/sybase/server10/install/KINGKONG10_errorlog'.
00:93/11/01 11:58:00.19 kernel  Using 256 file descriptors.
00:93/11/01 11:58:00.19 kernel  Network and device connection
limit is 253.
00:93/11/01 11:58:00.37 kernel  Initializing virtual device 0,
'/sybase/server10/devices/KINGKONG10_master.dat'
00:93/11/01 11:58:00.38 kernel  Virtual device 0 started using
standard unix i/o.
00:93/11/01 11:58:00.43 server  Number of buffers in buffer cache:
1668.
00:93/11/01 11:58:00.43 server  Number of proc buffers allocated:
417.
00:93/11/01 11:58:00.43 server  Number of blocks left for proc
headers: 393.
00:93/11/01 11:58:00.77 server  Opening Master Database ...
00:93/11/01 11:58:00.88 server  Loading SQL Server's default sort
order and character set
00:93/11/01 11:58:00.98 kernel  network name godzilla, type ether,
port 2351
00:93/11/01 11:58:01.06 server  Recovering database 'master'
00:93/11/01 11:58:01.07 server  Recovery dbid 1 ckpt (1052,30)
00:93/11/01 11:58:01.10 server  Recovery no active transactions
before ckpt.
00:93/11/01 11:58:01.19 server  server is unnamed
00:93/11/01 11:58:01.22 server  Activating disk 'sybsecurity'.
00:93/11/01 11:58:01.22 kernel  Initializing virtual device 2,
'/sybase/server10/devices/KINGKONG10_sec.dat'
00:93/11/01 11:58:01.22 kernel  Virtual device 2 started using
standard unix i/o.
00:93/11/01 11:58:01.22 server  Activating disk 'sysprocsdev'.
00:93/11/01 11:58:01.22 kernel  Initializing virtual device 1,
'/sybase/server10/devices/KINGKONG10_procs.dat'
00:93/11/01 11:58:01.22 kernel  Virtual device 1 started using
standard unix i/o.
00:93/11/01 11:58:01.28 server  Recovering database 'sybsecurity'.
```

```
00:93/11/01 11:58:01.29 server  Recovery dbid 5 ckpt (323,32)
oldest tran=(323,31)
00:93/11/01 11:58:01.32 server  1 transactions rolled forward.
00:93/11/01 11:58:01.42 server  audproc: Loading global audit
options from sysauditoptions.
00:93/11/01 11:58:01.42 server  audproc: Global audit options
successfully loaded.
00:93/11/01 11:58:01.44 server  Recovering database 'model'.
00:93/11/01 11:58:01.44 server  Recovery dbid 3 ckpt (320,13)
00:93/11/01 11:58:01.44 server  Recovery no active transactions
before ckpt.
00:93/11/01 11:58:01.46 server  Clearing temp db
00:93/11/01 11:58:02.21 server  Recovering database
'sybsystemprocs'.
00:93/11/01 11:58:02.96 server  Recovery dbid 4 ckpt (3631,26)
oldest tran=(3631,25)
00:93/11/01 11:58:07.11 server  19 transactions rolled forward.
00:93/11/01 11:58:08.67 server  Recovery complete.
00:93/11/01 11:58:08.75 server  SQL Server's default sort order
is:
00:93/11/01 11:58:08.78 server          'bin_iso_1' (ID = 50)
00:93/11/01 11:58:08.78 server  on top of default character set:
00:93/11/01 11:58:08.78 server          'iso_1' (ID = 1).
```

7. Check the value of DSQUERY (Unix: printenv DSQUERY; OpenVMW: SHOW DSQUERY)) and access that Server as *sa* using isql. The password is null.

To check the value of DSQUERY, use either:

```
godzilla% echo $DSQUERY
KINGKONG10
```

or:

```
godzilla% printenv DSQUERY
KINGKONG10
```

To access SQL Server:

```
godzilla% isql -Usa
Password:
1>
```

or:

```
godzilla% isql -Usa -P
1>
```

8. Run `select @@servername`. Then exit isql.

```
1> select @@servername
```

```
2> go

 ------------------------------
 KINGKONG10

(1 row affected)
```

9. What command would you execute to stop SQL Server?

While logged in to the server, type 'shutdown' e.g.

```
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
```

10. What command would you use to start SQL Server again?

```
godzilla% startserver -f RUN_KINGKONG10
```

Lab 3a - Initializing Devices

1. **Write and run a script which creates a disk device named *data_devN*, where *N* is your user number. The associated physical device should be a file called *data_devN.dat*. Make your device 4 MB. Use a *vdevno* assigned by the instructor.**
   Reminder: Size is in pages, 1 page = 2K, 1 512 pages = 1Mb.
   Note: Your device will be in the *sybase/install* directory. At your site, you would probably put devices in a separate directory.

Sample script file:

```
/* create the data_dev42 device */
disk init
name = "data_dev42",
physname = "/sybase/server10/devices/data_dev42.dat",
vdevno = 3,
size = 2048
go
```

To run the script:

```
godzilla% isql -Ustudent42 -Pstudent42 < data_dev42.script
```

2. **Log into SQL Server using isql and use a stored procedure to check your results. Exit isql.**

```
1> sp_helpdevice data_dev42
2> go
 device_name                    physical_name
        description


        status cntrltype device_number low         high
 ------------------------------
-----------------------------------------------

-------------------------------------------------------------------
------
-------------------------------------------------------------------
--------------
-------------------------------------------------------------------
--------------
------------------
        ------ --------- ------------ ---------- -----------
 data_dev42
/sybase/server10/devices/data_dev42.dat
        special, physical disk, 4.00 MB


             2         0             3    50331648    50333695

(1 row affected, return status = 0)
```

3. Write and run a script which makes the device which you just created a default disk. Check that your code had the desired effect.

Sample script file:

```
/* make data_dev42 a default device */
sp_diskdefault data_dev42, defaulton
go
```

To run the script:

```
godzilla% isql -Ustudent42 -Pstudent42 < data_dev42_default.script
(return status = 0)
```

Checking your results:

```
godzilla% isql -Ustudent42 -Pstudent42
1> sp_helpdevice data_dev42
2> go
 device_name                    physical_name
         description


         status cntrltype device_number low         high
 ------------------------------
-------------------------------------------

-------------------------------------------------------------------
------
-------------------------------------------------------------------
--------------
-------------------------------------------------------------------
--------------
------------------
        ------ --------- ------------ ----------- -----------
 data_dev42
/sybase/server10/devices/data_dev42.dat
         special, default disk, physical disk, 4.00 MB


              3         0             3    50331648    50333695

(1 row affected, return status = 0)
```

4. Check if the master device is a default disk. If so, change it so that master is not a default device. Why is this a good idea? Check your results.

```
1> sp_helpdevice master
2> go
 device_name                    physical_name
         description
```

```
        status cntrltype device_number low         high
 ------------------------------
-------------------------------------------

------------------------------------------------------------------
------
------------------------------------------------------------------
-------------
------------------------------------------------------------------
-------------
-----------------
       ------ --------- ------------ ---------- -----------
 master                          d_master
        special, default disk, physical disk, 17.00 MB


             3         0             0           0        8703

(1 row affected, return status = 0)


1> sp_diskdefault master, defaultoff
2> go
(return status = 0)
```

Turning off the default disk status for master will help to prevent the creation or alteration of user databases on the master device.

```
1> sp_helpdevice master
2> go
 device_name                    physical_name
        description


        status cntrltype device_number low         high
 ------------------------------
-------------------------------------------

------------------------------------------------------------------
------
------------------------------------------------------------------
-------------
------------------------------------------------------------------
-------------
-----------------
       ------ --------- ------------ ---------- -----------
 master                          d_master
        special, physical disk, 17.00 MB


             2         0             0           0        8703

(1 row affected, return status = 0)
```

5. Write and run a script file which creates a second disk device called *log_devN*. Make it 2 MB, and map it to *log_devN.dat*.

Sample script:

```
/* create the log_dev42 device */
disk init
name = "log_dev42",
physname = "/sybase/server10/devices/log_dev42.dat",
vdevno = 4,
size = 1024
go
```

To run the script:

```
godzilla% isql -Ustudent42 -Pstudent42 < log_dev42.script
```

6. (Optional) Write a query which given a logical device name will return the corresponding physical device name. Run sp_helpdevice to get some logical device names, and test your query.

```
1> sp_helpdevice
2> go
 device_name                    physical_name
        description


        status cntrltype device_number low         high
 ------------------------------
-----------------------------------------------

-------------------------------------------------------------------
------
-------------------------------------------------------------------
--------------
-------------------------------------------------------------------
--------------
------------------
        ------ --------- ------------- ----------- -----------
 data_dev42
/sybase/server10/devices/data_dev42.dat
        special, default disk, physical disk, 4.00 MB


             3         0             3    50331648    50333695
 log_dev42
/sybase/server10/devices/log_dev42.dat
        special, physical disk, 2.00 MB


             2         0             4    67108864    67109887
 master                          d_master
        special, physical disk, 17.00 MB
```

```
             2         0             0           0        8703
 sybsecurity
/sybase/server10/devices/KINGKONG10_sec.dat
         special, physical disk, 5.00 MB


             2         0             2    33554432    33556991
 sysprocsdev
/sybase/server10/devices/KINGKONG10_procs.dat
         special, physical disk, 10.00 MB


             2         0             1    16777216    16782335
 tapedump1                      /dev/rmt4
         tape,        625 MB, dump device


            16         3             0           0       20000
 tapedump2                      /dev/rst0
         disk, dump device


            16         2             0           0       20000

(7 rows affected, return status = 0)
1> select phyname from sysdevices where name = "log_dev42"
2> go

         phyname


------------------------------------------------------------------
------
------------------------------------------------------

         /sybase/server10/devices/log_dev42.dat


(1 row affected)
```

Lab 3b - Mirroring

1. Use a stored procedure to display details about *log_devN* (created in the last lab).

```
1> sp_helpdevice log_dev42
2> go
 device_name                    physical_name
        description


        status cntrltype device_number low         high
 ----------------------------
-------------------------------------------

-------------------------------------------------------------------
------
-------------------------------------------------------------------
--------------
-------------------------------------------------------------------
--------------
------------------
       ------ --------- ------------ ----------- -----------
 log_dev42
/sybase/server10/devices/log_dev42.dat
        special, physical disk, 2.00 MB


             2         0             4    67108864    67109887

(1 row affected, return status = 0)
```

Is the device mirrored?

No, it's not currently mirrored. (Mirroring would be mentioned in the description column.)

2. Activate mirroring for that device. Call the file which will mirror your device "log_dev_mirrorN.dat".

```
1> disk mirror
2> name = "log_dev42",
3> mirror = "/sybase/server10/devices/log_dev_mirror42.dat"
4> go
Creating the physical file for the mirror...
Starting Dynamic Mirroring of 1024 pages for logical device
'log_dev42'.
The remaining 1024 pages are currently unallocated and will be
mirrored as theyare allocated.
```

3. Use a stored procedure to check your results. Which is the primary device, and which is the secondary?

```
1> sp_helpdevice log_dev42
2> go
 device_name                    physical_name
         description


         status cntrltype device_number low         high
 ------------------------------
-----------------------------------------------

-----------------------------------------------------------------------
------
-----------------------------------------------------------------------
--------------
-----------------------------------------------------------------------
--------------
------------------
        ------ --------- ------------ ----------- -----------
 log_dev42
/sybase/server10/devices/log_dev42.dat
         special, MIRROR ENABLED, mirror =
'/sybase/server10/devices/log_dev_mirror42.dat', serial writes,
reads mirrored, physical disk, 2.00 MB


            738         0             4    67108864    67109887

(1 row affected, return status = 0)
```

Primary is log_dev42.dat, secondary is log_dev_mirror42.dat

4. Assume the following:
Because of resource constraints, you can mirror only three of the devices listed below. Which would you choose to mirror?

A_dev	**contains master database**
B_dev	**contains tempdb**
C_dev	**contains data from frequently-updated database1**
D_dev	**contains log from frequently-updated database1**
E_dev	**contains data from frequently-updated database2**
F_dev	**contains log from frequently-updated database2**

Mirror A_dev, D_dev and F_dev.

5. For the above example, what if C_dev and D_dev were used for a "quiet" database? Would your answer change?

Mirror A_dev, E_dev and F_dev.

Lab 4a - Creating Databases

1. Write and run a script in which you create a 3 MB database called *dbN*, where N is your user number. Place the data (2 MB) on *data_devN* and the log (1 MB) on *log_devN*.

Sample script:

```
/* create database db42 */
create database db42
on data_dev42 = 2
log on log_dev42 = 1
go
```

To run the script:

```
godzilla% isql -Ustudent42 -Pstudent42 < create_db42.script
CREATE DATABASE: allocating 1024 pages on disk 'data_dev42'
CREATE DATABASE: allocating 512 pages on disk 'log_dev42'
```

2. Using a stored procedure, display information about this database, including size, placement, ownership. Now find your database in *sysdatabases*.

```
1> sp_helpdb db42
2> go
 name                 db_size      owner                  dbid
        created
        status

 ---------------------- ------------ ----------------------
------
        --------------

-------------------------------------------------------------------
------
------------------------------
 db42                         3.0 MB student42                  7
        Nov 01, 1993
        no options set


 device_fragments              size          usage
free kbytes
 ----------------------------- ------------ --------------------
-----------
 data_dev42                    2.0 MB        data
only                    1376
 log_dev42                     1.0 MB        log
only                     1008

(return status = 0)
1> select * from sysdatabases where dbid = 7
2> go
 name                          dbid   suid   status version logptr
       crdate                   dumptrdate                status2
        audflags    deftabaud   defvwaud    defpraud
```

```
 ------------------------------ ------ ------ ------ -------
-----------
        -------------------------- --------------------------
-------
        ----------- ----------- ----------- -----------
 db42                                7      3      0       1        1026
              Nov  1 1993  4:46PM        Nov  1 1993  4:46PM        0
                   0            0           0            0

(1 row affected)
```

3. Display all the objects in your database. Besides system tables, what objects does the database contain? Can you influence what objects a newly-created database has? If so, how? (Don't do this, however!)

```
1> use db42
2> go
1> sp_help
2> go
 Name            Owner           Object_type
 --------------- --------------- ----------------------
 sysalternates   dbo             system table
 syscolumns      dbo             system table
 syscomments     dbo             system table
 sysconstraints  dbo             system table
 sysdepends      dbo             system table
 sysgams         dbo             system table
 sysindexes      dbo             system table
 syskeys         dbo             system table
 syslogs         dbo             system table
 sysobjects      dbo             system table
 sysprocedures   dbo             system table
 sysprotects     dbo             system table
 sysreferences   dbo             system table
 sysroles        dbo             system table
 syssegments     dbo             system table
 systhresholds   dbo             system table
 systypes        dbo             system table
 sysusermessages dbo             system table
 sysusers        dbo             system table

 User_type       Storage_type    Length Nulls Default_name
Rule_name
 --------------- --------------- ------ ----- ---------------
---------------

(return status = 0)
```

This shows that there are no other objects besides system tables. Yes, you can influence this; adding objects to model will result in those objects existing in subesequently created databases.

4. Make the database for dbo use only. Using a stored procedure, verify that your change took effect. Then undo this (set "dbo use only" back to false).

```
1> use master
2> go
1> sp_dboption db42, "dbo use only", true
2> go
Database option 'dbo use only' turned ON for database 'db42'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use db42
2> go
1> checkpoint
2> go
1> sp_helpdb db42
2> go
 name                 db_size      owner                 dbid
        created
        status

 -------------------- ------------ ----------------------
------
       --------------

-------------------------------------------------------------------
------
------------------------------
 db42                          3.0 MB student42                  7
        Nov 01, 1993
        dbo use only


 device_fragments              size         usage
free kbytes
 ----------------------------- ------------ --------------------
-----------
 data_dev42                    2.0 MB       data
only                    1376
 log_dev42                     1.0 MB       log
only                     1008

 device                        segment
 ----------------------------- ------------------------------
 data_dev42                    default
 data_dev42                    system
 log_dev42                     logsegment

(return status = 0)
1> use master
2> go
1> sp_dboption db42, "dbo use only", false
2> go
Database option 'dbo use only' turned OFF for database 'db42'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use db42
2> go
1> checkpoint
2> go
1> sp_helpdb db42
2> go
```

```
 name                 db_size      owner                 dbid
        created
        status

 -------------------- ------------ --------------------
------
       --------------

-----------------------------------------------------------------
------
------------------------------
 db42                          3.0 MB student42                 7
        Nov 01, 1993
        no options set


 device_fragments              size         usage
free kbytes
 ----------------------------- ------------ -------------------
-----------
 data_dev42                    2.0 MB       data
only                    1376
 log_dev42                     1.0 MB       log
only                     1008

 device                        segment
 ----------------------------- ------------------------------
 data_dev42                    default
 data_dev42                    system
 log_dev42                     logsegment

(return status = 0)
```

5. (Optional) Write a query to show which databases are on your device *data_devN*. Now run your query again for the master device. (Hint: Comparing *sysusages.vstart* with *sysdevices.low* and *sysdevices.high* will tell you which device each database fragment is on.)

```
1> use master
2> go
1> select distinct
2>      "physical device" = sysdevices.phyname,
3>      "database name" = sysdatabases.name
4> from sysdevices, sysdatabases, sysusages
5> where sysdevices.name = "data_dev42"
6> and sysusages.dbid = sysdatabases.dbid
7> and sysusages.vstart between sysdevices.low and sysdevices.high
8> go

         physical device

         database name


-----------------------------------------------------------------
------
-------------------------------------------------------
        ------------------------------
```

```
        /sybase/server10/devices/data_dev42.dat

        db42

(1 row affected)
```

Or, for tidier output:

```
1> select distinct
2>      "physical device" = convert(char(40), sysdevices.phyname),
3>      "database name" = convert(char(15), sysdatabases.name)
4> from sysdevices, sysdatabases, sysusages
5> where sysdevices.name = "data_dev42"
6> and sysusages.dbid = sysdatabases.dbid
7> and sysusages.vstart between sysdevices.low and sysdevices.high
8> go
 physical device                          database name
 ---------------------------------------- ---------------
 /sybase/server10/devices/data_dev42.d db42

(1 row affected)
```

And, for the master device:

```
1> use master
2> go
1> select distinct
2>      "physical device" = convert(char(40), sysdevices.phyname),
3>      "database name" = convert(char(15), sysdatabases.name)
4> from sysdevices, sysdatabases, sysusages
5> where sysdevices.name = "master"
6> and sysusages.dbid = sysdatabases.dbid
7> and sysusages.vstart between sysdevices.low and sysdevices.high
8> go
 physical device                          database name
 ---------------------------------------- ---------------
 d_master                                 master
 d_master                                 model
 d_master                                 pubs2
 d_master                                 tempdb

(4 rows affected)
```

6. (Optional) Write a query to determine how much free space there is on your device *data_devN*. (Hint: The relevant rows in sysusages can be found using the same comparison as in #5 above.)

```
1> select
2>      name,
3>      size = (d.high - d.low + 1),
4>      used = sum(u.size),
5>      left = (d.high - d.low + 1) - sum(u.size)
6> from
7>      sysdevices d, sysusages u
8> where
```

```
9>       name = "data_dev42"
10>       and u.vstart between d.low and d.high
11> group by
12>      name
13> go
 name                           size        used        left
 ------------------------------ ----------- ----------- -----------
 data_dev42                           2048        1024        1024

(1 row affected)
```

7. (Optional) Given the following output from `sp_helpdb sales`, write the commands which would recreate *sales* if it were lost:

```
name  db_size owner dbid  created      status
sales 4 MB    sa    5     Oct 16 1992 no options set

device_fragments  size    usage          free kbytes
dev1              2 MB    data only             1376
dev2              1 MB    log only              1008
dev3              1 MB    data only             1008
```

The commands would be:

```
create database sales on dev1 = 2 log on dev2 = 1
alter database sales on dev3 = 1
```

Lab 4b - Segments

1. Put the following commands in logical sequence first, and then execute them:

___ Create a non-clustered index for *mytable* on *seg1*
___ Add a segment called *seg1*, and place it on *index_devN*
___ Alter your database so part of it is on *index_devN*
___ Create a table called *mytable* on the default segment
___ Add a 1 MB device called *index_devN*
___ Drop *system* and *default* from *index_devN*

Sample create table statement:

```
create table mytable (a int, b int)
```

Use a unique *vdevno* (provided by instructor) for the disk init statement, and put your device on a file named *index_devN.dat*. Use `sp_helpsegment` to verify your results.

---

1. Add a 1Mb device called index_devN
2. Alter your database so part of it is on index_devN
3. Add a segment called seg1, and place it on index_devN
4. Drop system and default from index_devN
5. Create a table called mytable on the default segment
6. Create a non-clustered index for mytable on seg1

Now, the commands:

1. Add a 1Mb device called index_devN

```
1> disk init
2> name = "index_dev42",
3> physname = "/sybase/server10/devices/index_dev42.dat",
4> vdevno = 5,
5> size = 512
6> go
```

2. Alter your database so part of it is on index_devN. (sp_helpdb will show the result of this command.)

```
1> alter database db42 on index_dev42 = 1
2> go
Extending database by 512 pages on disk index_dev42
1> sp_helpdb db42
2> go
 name                    db_size      owner                    dbid
        created
        status

 ----------------------- ------------ -----------------------
------
        --------------
```

```
------------------------------------------------------------------
------
------------------------------
 db42                          4.0 MB student42                    7
        Nov 01, 1993
        no options set


 device_fragments               size          usage
free kbytes
 ------------------------------ ------------ --------------------
-----------
 data_dev42                     2.0 MB        data
only                    1376
 index_dev42                    1.0 MB        data
only                    1024
 log_dev42                      1.0 MB        log
only                     1008

(return status = 0)
```

3. Add a segment called seg1, and place it on index_devN.

```
1> use db42
2> go
1> sp_addsegment seg1, db42, index_dev42
2> go
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
Segment created.



(return status = 0)
```

4. Drop system and default from index_devN. (Remember that segments are per database; thus you have to 'use' your database before adding or dropping them. sp_helpdb will again show the results; note that because you are now using the database referred to by sp_helpdb, you will get detailed information about segments.)

```
1> sp_dropsegment system, db42, index_dev42
2> go
DBCC execution completed. If DBCC printed error messages, contact
a user with
System Administrator (SA) role.
Segment reference to device dropped.
(return status = 0)
1> sp_dropsegment "default", db42, index_dev42
2> go
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
Segment reference to device dropped.
(return status = 0)
1> sp_helpdb db42
2> go
 name                    db_size      owner                    dbid
        created
        status
```

```
 ------------------------ ------------ ------------------------
------
        --------------

------------------------------------------------------------------
------
------------------------------
 db42                          4.0 MB student42                   7
        Nov 01, 1993
        no options set


 device_fragments               size          usage
free kbytes
 ------------------------------ ------------ --------------------
-----------
 data_dev42                     2.0 MB        data
only                    1376
 index_dev42                    1.0 MB        data
only                    1024
 log_dev42                      1.0 MB        log
only                     1008

 device                         segment
 ------------------------------ ------------------------------
 data_dev42                     default
 data_dev42                     system
 index_dev42                    seg1
 log_dev42                      logsegment

(return status = 0)
```

5. Create a table called mytable on the default segment.

```
1> create table mytable(a int, b int)
2> go
```

6. Create a non-clustered index for mytable on seg1. (Use sp_helpsegment for both seg1, and for the default segment to see the placement of mytable and its index. Note that because "default" is a keyword, it must be in quotes when used as an argument to sp_helpsegment.)

```
1> create index mytable_index on mytable(a) on seg1
2> go
1> sp_helpsegment seg1
2> go
 segment name                          status
 ------- ------------------------------ ------
       3 seg1                                0

 device                         size                    free_pages
 ------------------------------ ---------------------- -----------
 index_dev42                    1.0MB                          504

 table_name                    index_name                    indid
 ------------------------------ ------------------------------
------
 mytable                       mytable_index                    2
```

```
(return status = 0)
1> sp_helpsegment "default"
2> go
 segment name                          status
 ------- ----------------------------- ------
       1 default                            1

 device                         size                  free_pages
 ------------------------------ --------------------- -----------
 data_dev42                     2.0MB                         680

 table_name                    index_name                   indid
 ------------------------------ ------------------------------
------
 mytable                       mytable                        0
 syscomments                   syscomments                    1
 sysusermessages               csysusermessages               1
 sysusermessages               ncsysusermessages              2

(return status = 0)
```

2. Assume the following: There has been a system failure, and for some reason you have no backups and no scripts. Fortunately, you have up-to-date hard copies of the system tables. Looking at the excerpts of *sysdevices*, *sysusages*, and *syssegments* on the next page, answer the following questions:

(a) How many device fragments are in the database whose *dbid* is 4?

There are 3 fragments.

(b) How big are these fragments?

Respectively 2560, 1024 and 1024 pages (or, 5Mb, 2Mb and 2Mb)

(c) Which devices are they on?

In order, data_dev, log_dev, contacts_dev

(d) What segments are mapped to each device?

On data_dev, system and default; on log_dev, logsegment; on contacts_dev, contacts_seg

3. (Optional) The rows in *sysusages* are in chronological order, according to how databases, etc. were created. Given that, determine what sequence of statements would be required to recreate the lost database.

Assuming the devices are intact:

Firstly, use master:

```
1. create database contacts on data_dev = 5 log on log_dev = 2
```

```
2. alter database contacts on contacts_dev = 2
```

Then, use contacts:

```
1. sp_addsegment contacts_seg, contacts, contacts_dev
2. sp_dropsegment system, contacts, contacts_dev
3. sp_dropsegment "default", contacts, contacts_dev
```

Lab 5a - Roles & Special Users

1. **Which roles are associated with the following tasks? Assume the defaults, i.e., no permissions explicitly granted. Where there is more than one correct answer, fill in all that apply.**

   Possible answers: SA, SSO, OPER, dbo, None of the above

Manage the audit system:	SSO
Add new server logins:	SSO
Back up a database:	OPER, dbo, SA
Create a table:	dbo, SA
Run dbcc in a database:	dbo, SA
Create a database:	SA
Install SQL Server:	None of the above.
Drop database users:	dbo, SA
Change login passwords:	SSO
Checkpoint a database:	dbo, SA
Access a table:	None of the above; dbo and SA via setuser.
Load a database:	OPER, dbo, SA
Allocate a dump device:	SA

Lab 5b - Logins, Users & Groups

1. Add three logins to SQL Server. Call them *tomN*, *dickN*, and *harryN* (where N is your user number), and make your database their default database. Give them all "friday" as a password.

```
1> sp_addlogin tom42, friday, db42
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
1> sp_addlogin dick42, friday, db42
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
1> sp_addlogin harry42, friday, db42
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
```

2. Add these logins as users in your database.

```
1> use db42
2> go
1> sp_adduser tom42
2> go
New user added.
(return status = 0)
1> sp_adduser dick42
2> go
New user added.
(return status = 0)
1> sp_adduser harry42
2> go
New user added.
(return status = 0)
```

3. *harryN* has forgotten his password and needs help. If you were a System Security Officer, what could you do? Write down the command you would use.

As an SSO, you could assign him a new password by executing:

```
sp_password caller_passwd, new_passwd, login_name
```

In this case, as the SSO with the password 'student42', the command to give the new password of 'saturday' to harry42 would be:

```
sp_password student42, saturday, harry42
```

4. Create a group called *programmers* and add users *tomN*, *dickN*, and *harryN* to that group. Execute `sp_helpgroup` and `sp_helpgroup programmers`, and note the difference in output.

```
1> sp_addgroup programmers
2> go
New group added.
(return status = 0)
1> sp_changegroup programmers, tom42
2> go
Group changed.
(return status = 0)
1> sp_changegroup programmers, dick42
2> go
Group changed.
(return status = 0)
1> sp_changegroup programmers, harry42
2> go
Group changed.
(return status = 0)
1> sp_helpgroup
2> go
 Group_name                     Group_id
 ------------------------------ --------
 navigator_role                    16388
 oper_role                         16386
 programmers                       16390
 public                                0
 replication_role                  16389
 sa_role                           16384
 sso_role                          16385
 sybase_ts_role                    16387

(8 rows affected, return status = 0)
1> sp_helpgroup programmers
2> go
 Group_name               Group_id Users_in_group            Userid
 ------------------------ -------- ------------------------ ------
 programmers                 16390 dick42                        4
 programmers                 16390 harry42                       5
 programmers                 16390 tom42                         3

(3 rows affected, return status = 0)
```

Lab 5c - Roles

1. Grant SA and OPER roles to *tomN*. Grant SSO role to *dickN*. Don't grant a role to *harryN*. Use `sp_displaylogin` to check role configuration. When will these take effect?

Granting roles:

```
1> sp_role "grant", "sa_role", tom42
2> go
Authorization updated.
(return status = 0)
1> sp_role "grant", "oper_role", tom42
2> go
Authorization updated.
(return status = 0)
1> sp_role "grant", "sso_role", dick42
2> go
Authorization updated.
(return status = 0)
```

Checking authorizations:

```
1> sp_displaylogin tom42
2> go
Suid: 4
Loginame: tom42
Fullname:
Configured Authorization: sa_role oper_role
Locked: NO
Date of Last Password Change: Nov  2 1993  2:24PM
(return status = 0)
1> sp_displaylogin dick42
2> go
Suid: 5
Loginame: dick42
Fullname:
Configured Authorization: sso_role
Locked: NO
Date of Last Password Change: Nov  2 1993  2:24PM
(return status = 0)
1> sp_displaylogin harry42
2> go
Suid: 6
Loginame: harry42
Fullname:
Configured Authorization:
Locked: NO
Date of Last Password Change: Nov  2 1993  2:25PM
(return status = 0)
```

The newly added roles will take effect the next time the user logs in.

2. Log in as *tomN* and check what roles are active for that login. Try to add the new login *lauraN*, with your database as the default. Were you successful? Why or why not?

```
godzilla% isql -Utom42 -Pfriday
1> sp_displaylogin
2> go
Suid: 4
Loginame: tom42
Fullname:
Configured Authorization: sa_role oper_role
Locked: NO
Date of Last Password Change: Nov  2 1993  2:24PM
(return status = 0)
1> sp_addlogin laura42, friday, db42
2> go
Msg 567, Level 16, State 1:
Server 'KINGKONG10', Procedure 'sp_addlogin', Line 49:
You must have the following role(s) to execute this
command/procedure:  'sso_role' .  Please contact a user with the
appropriate role for help.
(return status = 1)
```

As shown by the error message above, laura42 could not be added because tom42 is not configured as an SSO.

3. Log in as *dickN*, check what roles are active for that login, and add a new login, *lauraN*. Make your database her default database. (We will add her as a user in a later lab.)

```
godzilla% isql -Udick42 -Pfriday
1> sp_displaylogin
2> go
Suid: 5
Loginame: dick42
Fullname:
Configured Authorization: sso_role
Locked: NO
Date of Last Password Change: Nov  2 1993  2:24PM
(return status = 0)
1> sp_addlogin laura42, friday, db42
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
```

4. As *dickN*, disable the SSO role (Hint: use set) and try to add a new login, *bobN*. What happens?

```
1> set role "sso_role" off
2> go
1> sp_addlogin bob42, friday, db42
2> go
Msg 567, Level 16, State 1:
Server 'KINGKONG10', Procedure 'sp_addlogin', Line 49:
```

```
You must have the following role(s) to execute this
command/procedure:  'sso_role' .  Please contact a user with the
appropriate role for help.
(return status = 1)
```

Because the 'set role' command takes effect immediately, the user no longer has the correct authorization to run sp_addlogin.

5. (Optional) Using your user name to log into SQL Server, create a stored procedure in your database that only logins with the SSO role can execute. The procedure should print "Success!" if successful, or exit with an error message if an login that does not have the SSO role attempts to execute it. Test this out , then disable the SSO role for your login and verify that you can no longer run the procedure. Then enable SSO role again.

```
godzilla% isql -Ustudent42 -Pstudent42
1> create proc sso_only
2> as
3> if proc_role("sso_role")=0
4>      begin
5>     print "SSO authorization is required to run this procedure"
6>      return
7>      end
8> else
9>      print "Success!"
10> go
1> exec sso_only
2> go
Success!
(return status = 0)
1> set role "sso_role" off
2> go
1> exec sso_only
2> go
SSO authorization is required to run this procedure
(return status = 0)
1> set role "sso_role" on
2> go
```

Lab 5d - Grant & Revoke Privileges

1. As *tomN* (who is also SA/dbo), create a two-column table in your database and insert a few rows.

```
godzilla% isql -Utom42 -Pfriday
1> use db42
2> go
1> create table two_col(i int, c char(4))
2> go
1> insert two_col values(1, "abcd")
2> go 6
(1 row affected)
6 xacts:
```

When you follow "go" with a number, the batch is executed that number of times. In this case, 6 identical rows were added, hence the "6 xacts:"

2. Create a stored procedure that prints a brief message to the screen and then selects all rows from the table.

```
1> create proc two_col_proc
2> as
3> print "two_col rows:"
4> select * from two_col
5> go
1> exec two_col_proc
2> go
two_col rows:
 i           c
 ----------- ----
           1 abcd
           1 abcd
           1 abcd
           1 abcd
           1 abcd
           1 abcd

(6 rows affected, return status = 0)
```

3. Add *lauraN* as a user in your database. Then grant and revoke permissions so that:

a) all users except *lauraN* can create stored procedures in the database

First, add *lauraN* as a user in your database:

```
1> sp_adduser laura42
2> go
New user added.
(return status = 0)
```

Then arrange for everyone but *lauraN* to create stored procedures in the database:

```
1> grant create procedure to public
2> go
1> revoke create procedure from laura42
2> go
```

b) *dickN* can select, insert, and update rows in the table you created, but not delete

```
1> grant select, insert, update on two_col to dick42
2> go
```

c) *harryN* can execute your procedure and can grant this permission to others

```
1> grant execute on two_col_proc to harry42 with grant option
2> go
```

d) *lauraN* can select just the first column of the table

```
1> grant select on two_col(i) to laura42
2> go
```

e) all users in the *programmers* group can create tables

```
1> grant create table to programmers
2> go
```

4. Display protections for the table and stored procedure, and display permissions for specific users and groups.

```
1> sp_helprotect two_col
2> go
 grantor         grantee          type     action
         object          column     grantable
 --------------- --------------- --------
----------------------------
         --------------- ---------- ---------
 dbo             dick42           Grant    Insert
         two_col         All         FALSE
 dbo             dick42           Grant    Select
         two_col         All         FALSE
 dbo             dick42           Grant    Update
         two_col         All         FALSE
 dbo             laura42          Grant    Select
         two_col         i           FALSE

(return status = 0)
```

```
1> sp_helprotect dick42
2> go
 grantor         grantee         type     action
         object          column     grantable
 --------------- --------------- --------
----------------------------
         --------------- ---------- ---------
 dbo             dick42          Grant    Insert
         two_col         All        FALSE
 dbo             dick42          Grant    Select
         two_col         All        FALSE
 dbo             dick42          Grant    Update
         two_col         All        FALSE

(return status = 0)
1> sp_helprotect laura42
2> go
 grantor         grantee         type     action
         object          column     grantable
 --------------- --------------- --------
----------------------------
         --------------- ---------- ---------
 dbo             laura42         Grant    Select
         two_col         i          FALSE
 dbo             laura42         Revoke   Create Procedure
                         All        FALSE

(return status = 0)
1> sp_helprotect programmers
2> go
 grantor         grantee         type     action
         object          column     grantable
 --------------- --------------- --------
----------------------------
         --------------- ---------- ---------
 dbo             programmers     Grant    Create Table
                         All        FALSE

(return status = 0)
```

5. Can *harryN* execute your procedure? As he does not have permission to select from the table, how do you explain this?

```
godzilla% isql -Uharry42 -Pfriday
1> exec two_col_proc
2> go
two_col rows:
 i           c
 ----------- ----
           1 abcd
           1 abcd
           1 abcd
           1 abcd
           1 abcd
           1 abcd

(6 rows affected, return status = 0)
```

Harry has execute permission for the stored procedure. As the underlying table is owned by the same user as the procedure itself, SQL Server does not check the permissions on the table, and the execute permission on the procedure selecting from the table is sufficient.

6. What command(s) would you use to ensure that only System Security Officers could run `sp_helpuser`? (Don't do this, however.)

In sybsystemprocs, you would execute:

```
revoke execute on sp_helpuser from public
grant execute on sp_helpuser to sso_role
```

Lab 6a -Configuration

1. Use a stored procedure to display configuration values.

```
godzilla% isql -Ustudent42 -Pstudent42
1> sp_configure
2> go
 name                         minimum     maximum     config_value
run_value
 ---------------------------- ----------- ----------- ------------
-----------
 recovery interval                  1       32767           0
5
 allow updates                      0           1           0
0
 user connections                   5  2147483647           0
25
 memory                          3850  2147483647           0
5120
 open databases                     5  2147483647           0
10
 locks                           5000  2147483647           0
5000
 open objects                     100  2147483647           0
500
 procedure cache                    1          99           0
20
 fill factor                        0         100           0
0
 time slice                        50        1000           0
100
 database size                      2       10000           0
2
 tape retention                     0         365           0
0
 recovery flags                     0           1           0
0
 nested triggers                    0           1           1
1
 devices                            4         256           0
10
 remote access                      0           1           1
1
 remote logins                      0  2147483647           0
20
 remote sites                       0  2147483647           0
10
 remote connections                 0  2147483647           0
20
 pre-read packets                   0  2147483647           0
3
 upgrade version                    0  2147483647        1000
1000
 default sortorder id               0         255          50
50
 default language                   0  2147483647           0
0
 language in cache                  3         100           3
3
```

```
 max online engines                   1          32           1
1
 min online engines                   1          32           1
1
 engine adjust interval               1          32           0
0
 cpu flush                            1  2147483647         200
200
 i/o flush                            1  2147483647        1000
1000
 default character set id             0         255           1
1
 stack size                       20480  2147483647           0
28672
 password expiration interval         0       32767           0
0
 audit queue size                     1       65535         100
100
 additional netmem                    0  2147483647           0
0
 default network packet size        512      524288           0
512
 maximum network packet size        512      524288           0
512
 extent i/o buffers                   0  2147483647           0
0
 identity burning set factor          1     9999999        5000
5000

(38 rows affected, return status = 0)
```

2. Are there any zeros (0) in the config_value column? What does this mean?

A zero in the config_value column indicates the default configuration value for the current platform.

3. Which values are currently in effect? Which will take effect after the next reconfigure/restart?

The values currently in effect are those in the run_value column. Any values in the config_value column which differ from those in the corresponding run_value column will take effect after the next reconfigure or restart, depending on whether they are dynamic options or not.

4. Given the current configuration, what effect would there be on the procedure and data caches if you increased the number of user connections by 8? (Predict the number of pages you will be losing in each.)

Allowing 50k per user connection, or 25 pages, 8 user connections will require 200 pages. As the procedure cache percentage is 20% this will result in a reduction there of 40 pages, and a reduction of 160 pages in the data cache.

5. You encounter the following problems. Assuming the problem to be one of configuration, how would you fix it?

a) Lots of physical reads (via set statistics io on)

High numbers of physical reads indicate that the data cache is inadequately sized, and that server memory should be increased.

b) Page fault count goes up

Increased page faulting indicates that the server memory is too large for the current operating system configuration. Either reconfigure the operating system appropriately, or reduce the amount of memory configured for the SQL Server.

c) User can't connect

If a user can't connect, the first thing to check is that the server is configured for sufficient user connections. The SQL Server errorlog will indicate if this is the resource that is being exhausted, with a message like this:

```
00:93/11/02 08:48:28.28 server  Error: 1601, Severity: 21, State:
3
00:93/11/02 08:48:28.28 server  There are not enough 'user
connections' available to start a new process. Retry when there are
fewer active users, or ask your System Administrator to
reconfigure SQL Server with more user connections.
```

d) "Msg 701: There is not enough procedure cache to run this procedure, trigger, or SQL batch. Retry later, or ask your SA to reconfigure SQL Server with more procedure cache."

This message indicates that there is not enough procedure cache available to run this query. More memory is required, either:

1) increase the overall memory for the server

2) increase the procedure cache size

3) make more memory available by reducing amount required for other resources, like unneeded user connections

6. Assume the following:

Total Mem = 64 Mb, OS needs 2 Mb, Misc. programs need 2 Mb
Three database devices to be mirrored, eight devices total
There are 25 SQL Server logins, with only 10 logged in to SQL Server at any one time
Of these, expect 2 application developers concurrently, each using a single

connection
Expect 8 application users
Application uses 3 connections
Application uses two stored procedures (10K, 30K) heavily.
Expect all stored procedures to be used concurrently by all application users

Based on these specifications, what are the minimal and maximal configurations for the following options:

memory
user connections (50K per user connection)
devices (512 bytes per device)
procedure cache

How much does each set of configurations leave for data cache?

The simplest way to approach this configuration is to determine what the memory requirements of each of the configurable resources are, and see how much that leaves available for cache. Here, we have potentially 60Mb to use (64Mb, minus 2Mb for the OS, and 2Mb for other programs). It's easiest to perform the calculation in units of 2k pages, so we have 30720 pages available.

Firstly, 8 devices at 512 bytes require 2 pages. Secondly, user connections. The number of logins is not relevant; what is, is the number of user connections required to support them. Here, it's 8 application users, at 3 connections each, and 2 developers with one connection each, making a total of 26. Round it to 30, (to give some headroom), at 50k or 25 pages each, gives 750 pages.

Each application user will require 10k + 30K of procedure cache, i.e. 40k, or 20 pages, which is 160 pages for 8 users. If we assume a 20% procedure cache size (the default), this predicts a data cache size of 640 pages.

Summing up what we have so far, in 2K pages:

Devices:	2
Connections:	750
Procedure cache:	160
Data cache:	640
Subtotal:	1552
Server executable	1400
Total:	2952

There are other factors to be taken into account here, but this example configuration could easily run on a SQL Server configured with the default amount of memory, of 3850 pages; this would be the minimal recommended configuration. Beyond that, memory could be allocated up to the maximum available of around 30,000 pages; clearly this system can support a much more sizeable configuration than that specified here

A more exhaustive treatment of the subject can be found in the *Sybase Troubleshooting Guide.*

Lab 7a - Transactions & Recovery

1. Assume there is a system failure. The *syslogs* table has the following recorded:

```
begin tran
   delete table1 where...
   insert table2 values (...)
```

Data on the disk has the `delete` but not the `insert`. What will SQL Server do upon recovery? Which changes will be reflected in the database, which will not?

As the transaction was not committed, the delete will be rolled back, and both tables will look as they did before the begin tran statement.

2. For each of the following sets of commands, predict what would happen if SQL Server were shut down and then brought back up. What would be rolled forward or rolled back? Would the inserted rows be in the appropriate tables?

Student A:
```
begin transaction a
insert tableA values (...)
commit transaction
go
```

As the transaction was committed, the new row will be in the table.

Student B:
```
begin transaction b
insert tableB values (...)
go
```

As the transaction was not committed, the new row will not be in the table. It may have been copied to disk by SQL Server before the Server was taken down, but in any case the transaction will be rolled back.

Student C:
```
begin transaction c
insert tableC values (...)
checkpoint
go
```

As a checkpoint occurred, the row will have been written to disk. As there was no commit statement, however, the transaction will be rolled back, and the row will be removed.

Student D:
```
begin transaction d
insert tableD values (...)
commit transaction
go
checkpoint
```

As the transaction was committed, the new row will be in the table. The Server will not have to copy the new row to disk since the checkpoint would have done that. There will be no "roll forward" statement.

Lab 8a - Adding Dump Devices

1. **Create a disk dump device named *dump_devN*, where N is your user number. The physical device should be a file called *dump_devN*.dat.**

```
1> sp_addumpdevice "disk", "dump_dev42",
2> "/sybase/server10/devices/dump_dev42.dat", 2
3> go
'Disk' device added.
(return status = 0)
```

2. **Create a second dump device named *logdump_devN*. The physical device should be a file called *logdump_devN.dat*.**

```
1> sp_addumpdevice "disk", "logdump_dev42",
2> "/sybase/server10/devices/logdump_dev42.dat", 2
3> go
'Disk' device added.
(return status = 0)
```

3. **Assume the following situation: You plan to use a local Backup Server called "B_LOCAL" and a remote Backup Server called "B_TUNA" on a machine called "tuna". What commands would you use to make these Backup Servers known to SQL Server? (Do not execute these commands, as we do not have Backup Servers by that name on our system.)**

Local Backup Server:

Supply the name at installation, or add it manually to both the interfaces file and sysservers. For the latter, use sp_addserver:

```
1> sp_addserver SYB_BACKUP, null, B_LOCAL
2> go
```

Remote Backup Server:

All that is required is to add an appropriate entry to the interfaces file for the remote Backup Server, B_TUNA. Then you can use the remote Backup Server name in your dump command. (When specifying the dump device, use "at *BackupServerName*".)

4. **(Optional) It's your first day on the job. The former System Administrator has quit without leaving any notes.**
   a) **What is the network name of the local Backup Server?**

Execute sp_helpserver and look at the Server network name associated with SYB_BACKUP.

```
godzilla% isql -Ustudent42 -Pstudent42
1> sp_helpserver
2> go
```

```
 name                                network_name
         status


         id
 ------------------------------ ------------------------------

-------------------------------------------------------------------
------
-------------------------------------------------------------------
--------------
-------------------------------------------------------------------
--------------
------------------------
        -----
 KINGKONG10                          KINGKONG10



         0
 SYB_BACKUP                          KINGKONG10_BACKUP
         timeouts, no net password encryption


         1

(return status = 0)
```

The network name of the Backup Server is KINGKONG10_BACKUP.

4. (continued)

b) Can you tell if SQL Server knows about a remote Backup Server? If so, what is its name, and where is it located?

This is a trick question. The remote Backup Server name is only required to be in the interfaces file, but there is no "marker" to identify it as a Backup Server. If the name of the remote Backup Server is not distinctive, you will not be able to tell it apart from any other Server.

Lab 8b - Dump Database

1. **If you do not already have a table called *sm_table*, then create one. Then:**
   a. **Insert a row.**
   b. **Dump the database to your dump device dump_devN.**
   c. **Insert another row, and display the rows of the table.**

```
godzilla% isql -Ustudent42 -Pstudent42
1> use db42
2> go
1> create table sm_table (x int, y char(20))
2> go
1> insert sm_table values (1, "first row")
2> go
(1 row affected)
1> dump database db42 to dump_dev42
2> go
Backup Server session id is:  6.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file
/sybase/server10/devices/dump_dev42.dat.
Backup Server: 6.28.1.1: Dumpfile name 'db42933060E753   ' section
number 0001
mounted on disk file '/sybase/server10/devices/dump_dev42.dat'
Backup Server: 4.58.1.1: Database db42: 276 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database db42: 284 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database db42).
1> insert sm_table values (2, "second row")
2> go
(1 row affected)
1> select * from sm_table
2> go
 x           y
 ----------- --------------------
           1 first row
           2 second row

(2 rows affected)
```

2. **Load the database from the dump device. Use the database, select from the table, and verify that the first row is there, the second is not.**

```
1> use master
2> go
1> load database db42 from dump_dev42
2> go
Backup Server session id is:  9.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db42933060E753   ' section
number 0001
```

```
mounted on disk file '/sybase/server10/devices/dump_dev42.dat'
Backup Server: 4.58.1.1: Database db42: 4100 kilobytes LOADed.
Backup Server: 4.58.1.1: Database db42: 4108 kilobytes LOADed.
Backup Server: 3.42.1.1: LOAD is complete (database db42).
Remirroring the affected portions of the usage map that are on
mirrored devices.
1> use db42
2> go
1> select * from sm_table
2> go
 x           y
 ----------- --------------------
           1 first row

(1 row affected)
```

3. Assume the following: the database *salesdb* has been backed up on the tape *public_dev* and is still mounted. There is enough space left on the tape for dumps of *customerdb* and *employeedb*, but no more. What command(s) would you use to dump those two databases onto the same tape as *salesdb* without overwriting the *salesdb* information? What command(s) would you use to load *employeedb*?

To dump customerdb and employeedb onto public_dev after salesdb:

```
1> dump database customerdb to public_dev
2> go
1> dump database employeedb to public_dev with unload
2> go
```

To load employeedb:

```
1> load database employeedb from public_dev
2> go
```

4. You have a large database, *hugedb*, that will not fit on a single dump device. You decide to split it across three dump devices: *dump1_dev, dump2_dev, dump3_dev*. What command(s) would you use to dump the database onto those devices?

```
1> dump database hugedb to dump1_dev
2> stripe on dump2_dev
3> stripe on dump3_dev
4> go
```

Lab 8c - Dump Transaction

1. **In your database, in *sm_table*:**
   a. **Insert a second row.**
   c. **Dump the database to your dump device *dump_devN*.**
   d. **Insert a third row.**
   e. **Dump the transaction log to your device *logdump_devN*.**

```
1> use db42
2> go
1> insert sm_table values (2, "second row")
2> go
(1 row affected)
1> use master
2> go
1> dump database db42 to dump_dev42
2> go
Backup Server session id is:  12.  Use this value when executing
the 'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db42933060E85A   ' section
number 0001 mounted on disk file
'/sybase/server10/devices/dump_dev42.dat'
Backup Server: 4.58.1.1: Database db42: 276 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database db42: 284 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database db42).
1> use db42
2> go
1> insert sm_table values (3, "third row")
2> go
(1 row affected)
1> dump tran db42 to logdump_dev42
2> go
Backup Server session id is:  59.  Use this value when executing
the 'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db42932240FB18   ' section
number 0001 mounted on disk file
'/sybase/server10/devices/logdump_dev42.dat'
Backup Server: 4.58.1.1: Database db42: 36 allocated kilobytes
DUMPed, equal to 75% of reserved space.
Backup Server: 4.58.1.1: Database db42: 40 allocated kilobytes
DUMPed, equal to 83% of reserved space.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database db42: 44 allocated kilobytes
DUMPed, equal to 91% of reserved space.
Backup Server: 3.42.1.1: DUMP is complete (database db42).
```

2. Load the database from *dump_devN*. Which rows are there?

```
1> use master
2> go
1> load database db42 from dump_dev42
2> go
Backup Server session id is:  56.  Use this value when executing
the 'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db42932240F9E8   ' section
number 0001
mounted on disk file '/sybase/server10/devices/dump_dev.dat'
Backup Server: 4.58.1.1: Database db42: 4100 allocated kilobytes
LOADed, equal to 100% of reserved space.
Backup Server: 4.58.1.1: Database db42: 4108 allocated kilobytes
LOADed, equal to 100% of reserved space.
Backup Server: 3.42.1.1: LOAD is complete (database db42).
Remirroring the affected portions of the usage map that are on
mirrored devices.
1> use db42
2> go
1> select * from sm_table
2> go
 x           y
 ----------- --------------------
           1 first row
           2 second row

(2 rows affected)
```

The first two rows are there, but not the third. Load database overwrote the data and did not have a record of the third row.

3. Load the transaction dump from *logdump_devN*. Which rows are there now?

```
1> use master
2> go
1> load tran db42 from logdump_dev42
2> go
Backup Server session id is:  64.  Use this value when executing
the
'sp_volchanged' system stored procedure after fulfilling any
volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db42932240FB18   ' section
number 0001
mounted on disk file '/sybase/server10/devices/logdump_dev42.dat'
Backup Server: 4.58.1.1: Database db42: 44 allocated kilobytes
LOADed, equal to
1% of reserved space.
Backup Server: 3.42.1.1: LOAD is complete (database db42).
Remirroring the affected portions of the usage map that are on
mirrored devices.
Roll forward transaction 'ins'.
Roll back transaction 'dmpxact' -- no 'end transaction'.
1 transactions rolled forward.
```

```
1 transactions rolled back.
1> use db42
2> go
1> select * from sm_table
2> go
 x           y
 ----------- --------------------
           1 first row
           2 second row
           3 third row

(3 rows affected)
```

The third row is back to stay, having been inserted when SQL Server loaded the transaction log dump.

4. **Create a last-chance threshold procedure called *sp_thresholdaction* in your own database, to be executed when your log segment's last-chance threshold is crossed. The procedure should dump the transaction log to logdump_devN and print a message to the error log. Test that your procedure works by running an insert/delete batch (see below) terminated by the isql "go 3000", which causes it to execute 3000 times. Check the error log to see if your threshold procedure writes a message.**
   batch: insert sm_table values (...)
   delete sm_table
   go 3000
   (We suggest an insert followed by a delete so the data segment doesn't fill up before the log....)

Create threshold procedure:

```
1> create proc sp_thresholdaction
2>      @dbname varchar(30),
3>      @segmentname varchar(30),
4>      @space_left int,
5>      @status int
6> as
7> dump transaction @dbname to logdump_dev42
8> print "LOG DUMP:  '%1!' for '%2!' dumped", @segmentname, @dbname
9> go
```

Test it out:

```
1> insert sm_table values (1, "I will not be here long...")
2> delete sm_table
3> go 3000
Space available in the log segment has fallen critically low in
database 'db42'.
 All future modifications to this database will be suspended until
the log is successfully dumped and space becomes available.
(1 row affected)
(1 row affected)
3000 xacts:
```

Check errorlog for print statement:

```
00:93/11/03 09:31:01.18 server  background task message: LOG DUMP:
'logsegment' for 'db42' dumped
```

Lab 9a - Monitoring Space & Using Thresholds

1. Examine the error log and determine the following:

When was SQL Server last started?

Check most recent "Recovery complete" message:

```
00:93/11/01 16:51:43.70 server  Recovery complete.
```

What is its default sort order?

```
00:93/11/01 16:51:43.74 server  SQL Server's default sort order
is:
00:93/11/01 16:51:43.74 server          'bin_iso_1' (ID = 50)
```

Did any errors occur during startup?

Check error log.

Have any errors occurred since startup?

Check error log.

2. Use `sp_spaceused` to determine the space used for the table *sm_table* in your database.

```
1> sp_spaceused sm_table
2> go
 name                 rowtotal    reserved   data       index_size
unused
 -------------------- ----------- ---------- ---------- ----------
----------
 sm_table            1           16 KB      2 KB       0 KB       14
KB

(return status = 0)
```

3. Use the `data_pgs( )` function to determine the space used for the log in your database.

```
1> use db42
2> go
1> select data_pgs(8, doampg) from sysindexes where id = 8
2> go

 -----------
         247
```

4. Use `sp_helpdb` and `sp_helpsegment` to determine how much free space is left for the log.

```
1> sp_helpdb db42
2> go
 name                 db_size      owner                 dbid
        created
        status

 ---------------------- ------------ ----------------------
------
       --------------

-------------------------------------------------------------------
-------
------------------------------
 db42                          4.0 MB student42
7
         Nov 01, 1993
         no options set


 device_fragments              size         usage
free kbytes
 ------------------------------ ------------ --------------------
-----------
 data_dev42                     2.0 MB       data
only                  1328
 index_dev42                    1.0 MB       data
only                  1008
 log_dev42                      1.0 MB       log
only                   992

 device                         segment
 ------------------------------ ------------------------------
 data_dev42                     default
 data_dev42                     system
 index_dev42                    seg1
 log_dev42                      logsegment

(return status = 0)
1> sp_helpsegment logsegment
2> go
 segment name                          status
 ------- ------------------------------ ------
       2 logsegment                          0

 device                         size                  free_pages
 ------------------------------ --------------------- -----------
 log_dev42                      1.0MB                         496

 table_name                    index_name                  indid
 ------------------------------ ------------------------------
------
 syslogs                       syslogs                       0

(return status = 0)
```

5. Verify your results using `dbcc checktable`. Comment on any differences.

```
1> dbcc checktable(syslogs)
2> go
Checking syslogs
The total number of data pages in this table is 12.
*** NOTICE:  Space used on the log segment is 0.02 Mbytes, 2.34%.
*** NOTICE:  Space free on the log segment is 0.98 Mbytes, 97.66%.
Table has 276 data rows.
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
```

Comments on the results of the preceding commands: These commands give slightly different results because they are working from different basic data. `dbcc checktable` is the most accurate, and `data_pgs()` is the least accurate. Immediately after `dbcc checktable`, however, `data_pgs()` is accurate since `dbcc checktable` updates the tables this function uses for its estimates.

6. (Optional) Set up a free-space threshold on your log segment that will cause a threshold procedure you write to be executed when the log is half full. (Refer to results above for total log size.) That threshold procedure should dump the log with truncate_only and write a message to the error log indicating the amount of space left in the log.
Test that your threshold procedure works by running an insert/delete batch similar to the one you ran in an earlier lab (see below). Check the error log to see if your threshold procedure generates an error message.
batch: insert sm_table values (...)
delete sm_table
go 2000
(We suggest a dump log with truncate_only to simplify things here. You would normally want to dump to a device!)

Create the threshold procedure:

```
1> create procedure halffull_proc
2>      @dbname varchar(30),
3>      @segmentname varchar(30),
4>      @space_left int,
5>      @status int
6> as
7> dump transaction @dbname with truncate_only
8> print "LOG DUMP:  '%1!' for '%2!' dumped,
9>      with '%3!' pages free",
10> @segmentname, @dbname, @space_left
11> go
```

Create the threshold, and associate the above procedure to it:

```
1> sp_addthreshold db42, logsegment, 256, halffull_proc
2> go
Adding threshold for segment 'logsegment' at '256' pages.
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
(return status = 0)
```

Test out your threshold and associated procedure:

```
1> insert sm_table values (1, "hello and goodbye")
2> delete sm_table
3> go 2000
(1 row affected)
(1 row affected)
2000 xacts:
```

Check errorlog for your print statement:

```
00:93/11/03 09:50:14.87 server  background task message: LOG DUMP:
'logsegment' for 'db42' dumped, with '256' pages free
```

Lab 9b - Monitoring

1. Of the following monitoring tools:

system error log	**dbcc checkdb**
OS monitoring programs	**dbcc checktable**
SQL Server error log	**dbcc checkcatalog**
Backup Server error log	**dbcc checkalloc**
threshold manager	**dbcc tablealloc**
sp_helpdb	**dbcc indexalloc**
sp_helpsegment	**dbcc memusage**
sp_spaceused	**select data_pgs( )**
select rowcnt( )	**sp_monitor**

which tool(s) would you use to monitor/detect the following:

whether the system tables are consistent

---

dbcc checkcatalog

how busy SQL Server's CPU has been

OS monitoring tools (such as 'ps' for Unix, or 'show system' for OpenVMS); also sp_monitor

space in transaction log

Any one of sp_helpdb, sp_helpsegment, the data_pgs() function or dbcc checktable(syslogs) can show space usage for the log. The threshold manager can be used to automate the process.

I/O failures

Usually the first place these will be apparent is in the SQL Server errorlog; the system errorlog may contain more information.

whether there is a structural integrity problem in the database

---

dbcc checkdb

whether there is a structural integrity problem in a specific table

---

dbcc checktable

whether enough memory has been allocated

OS monitoring programs

1. (continued) Which tool(s) would you use to monitor/detect the following:

hardware problems

System error log; also SQL Server errorlog may contain information that would lead you to suspect a hardware problem, such as an I/O failure.

which stored procedures have been used recently

dbcc memusage

whether pages in the database are correctly allocated

dbcc checkalloc for the whole database; dbcc tablealloc and indexalloc for individual tables and indexes.

2. Check the consistency of your database.

```
1> use db42
2> go
1> dbcc checkdb
2> go
Checking current database
Checking 1
The total number of data pages in this table is 2.
Table has 26 data rows.
Checking 2
The total number of data pages in this table is 2.
Table has 31 data rows.
Checking 3
The total number of data pages in this table is 5.
Table has 205 data rows.
Checking 4
The total number of data pages in this table is 1.
Table has 28 data rows.
Checking 5
The total number of data pages in this table is 8.
Table has 50 data rows.
Checking 6
The total number of data pages in this table is 1.
Table has 5 data rows.
Checking 7
The total number of data pages in this table is 1.
Table has 4 data rows.
Checking 8
The total number of data pages in this table is 147.
*** NOTICE:  Space used on the log segment is 0.29 Mbytes, 28.71%.
*** NOTICE:  Space free on the log segment is 0.71 Mbytes, 71.29%.
Table has 5460 data rows.
.
.
.
Checking 128003487
The total number of data pages in this table is 1.
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
```

3. Check page allocation in your database.

```
1> dbcc checkalloc
2> go
Checking current database
Database 'db42' is not in single user mode - may find spurious
allocation problems due to transactions in progress.
************************************************************
TABLE: sysobjects                OBJID = 1
INDID=1  FIRST=1          ROOT=8  SORT=0
        Data level: 1.  2 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=16         ROOT=16         SORT=0
        Indid     : 2.  1 Index Pages in 1 extents.
TOTAL # of extents = 3
************************************************************
TABLE: sysindexes                OBJID = 2
INDID=1  FIRST=24         ROOT=32         SORT=0
        Data level: 1.  2 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
TOTAL # of extents = 2
************************************************************
.
.
.
************************************************************
TABLE: sm_table          OBJID = 128003487
INDID=0  FIRST=321        ROOT=321        SORT=0
        Data level: 0.  1 Data  Pages in 1 extents.
TOTAL # of extents = 1
************************************************************
Processed 30 entries in the sysindexes for dbid 7.
Alloc page 0 (# of extent=32 used pages=83 ref pages=75)
Alloc page 256 (# of extent=14 used pages=28 ref pages=28)
Alloc page 512 (# of extent=1 used pages=1 ref pages=1)
Alloc page 768 (# of extent=1 used pages=1 ref pages=1)
Alloc page 1024 (# of extent=2 used pages=9 ref pages=9)
Alloc page 1280 (# of extent=18 used pages=144 ref pages=140)
Alloc page 1536 (# of extent=1 used pages=3 ref pages=3)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Total (# of extent=70 used pages=270 ref pages=258) in this
database
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
```

4. Check consistency of system tables in your database.

```
1> dbcc checkcatalog
2> go
Checking current database
The following segments have been defined for database 7 (database
name db42).
virtual start addr      size      segments
------------------      ------    -------------------------
50331648                      1024
                                     0
                                     1
67108864                      512
                                     2
```

```
83886080                        512
                                                3
DBCC execution completed. If DBCC printed error messages, contact
a user with System Administrator (SA) role.
```

Lab 9c- Troubleshooting (Discussion Lab)

Note: the answers presented here are by no means comprehensive, but intended to provide a couple of common causes for each of the stated problem scenarios. Also, in most cases the error message the user receives will contain sufficient information to allow complete diagnosis; here we've attempted to present some cases which may not be so obvious.

1. **For each of the following situations,**
   - what the problem might be
   - how you would investigate further
   - how to fix the problem.

   (a) SQL Server won't start.

It may already be running! Use showserver to check.

It may have been inappropriately reconfigured. For example, dramatically increasing the number of user connections without increasing the server's memory to support the increase, can, in severe cases, prevent the server from booting. In this case, using buildmaster with the -r option (Unix) or /reconfigure (Open VMS) will reset all the configuration variables back to the defaults and allow the server to boot.

(b) User "tom", a valid user in the accounting database, cannot access that database.

The database may not have been recovered yet. Users cannot access a database until recovery is complete. Examine the SQL Server errorlog to check on the progress of recovery, and try again later.

The database may be in the process of loading. In that case, wait.

The database may be inconsistent. Run dbcc.

(c) User "jill" cannot run a query she has run successfully many times.

There may be a permissions problem, or inconsistencies in the database.

She may be blocked by another process. Run sp_who and sp_lock.

(d) Queries that typically take 2 minutes to run are taking 10 minutes to run.

There may have been a significant change in the data in one or more of the tables referenced by the query, resulting in the index statistics not reflecting the data correctly. Running 'update statistics' after such changes is usually necessary.

Another user may be monopolizing the system. Use sp_who and sp_lock to see if users are being blocked, or if someone is using excessive resources.

Heavy auditing may be slowing down the system.

(e) A user complains that her application suddenly died.

Check for failure of client machine, server machine, client program, server program, network, or disks.

(f) Only select statements work in the salesdb database.

The database may be in 'read only' mode. Use sp_helpdb to verify.

The transaction log may be full. This would prevent any logged activity from taking place until space is freed up, while allowing selects to continue.

Lab 10a - Auditing

1. What command(s) would you use to install the audit system?

sybinit

2. Check to see if an auditing system has been installed on the SQL Server you are using. If so, what is the value of "audit queue size"?

Use sp_helpdb to display all the databases on your system, and look for one called sybsecurity.

```
1> sp_helpdb
2> go
 name                 db_size      owner                 dbid
        created
        status

 ---------------------- ------------ ----------------------
------
       --------------

-----------------------------------------------------------------
------
------------------------------
 db42                        4.0 MB student42               7
        Nov 01, 1993
        no options set

 master                      3.0 MB sa                      1
        Jan 01, 1900
        no options set

 model                       2.0 MB sa                      3
        Jan 01, 1900
        no options set

 pubs2                       2.0 MB sa                      6
        Nov 01, 1993
        no options set

 sybsecurity                 5.0 MB sa                      5
        Nov 01, 1993
        trunc log on chkpt

 sybsystemprocs             10.0 MB sa                      4
        Nov 01, 1993
        trunc log on chkpt

 tempdb                      2.0 MB sa                      2
        Nov 01, 1993
        select into/bulkcopy


(return status = 0)
```

Use sp_help in sybsystemprocs and look for the auditing stored procedures.

To display audit queue size:

```
1> sp_configure "audit queue size"
2> go
 name                          minimum     maximum     config_value
run_value
 ----------------------------- ----------- ----------- ------------
-----------
 audit queue size                    1       65535         100
100

(return status = 0)
```

3. Enable auditing of the following events:

a) All commands submitted by your neighbor (or someone working on your Server).

```
1> sp_auditlogin "student43", "cmdtext", "on"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

b) All attempts to "use" your database.

```
1> sp_auditdatabase db42, "both", "u"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

c) Attempts to execute procedures in your database.

```
1> use db42
2> go
1> sp_auditsproc "all", db42, "both"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

d) All attempts to insert into the *sm_table* table in your database.

```
1> sp_auditobject sm_table, db42, "both", "i"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

4. Determine if Server-wide auditing is set. If it is not, turn it on and verify that it is set.

```
1> sp_auditoption
2> go
 name                           sval
 ------------------------------ ------------------------------
 enable auditing                off
 logins                         off
 logouts                        off
 server boots                   off
 rpc connections                off
 roles                          off
 sa commands                    off
 sso commands                   off
 oper commands                  off
 navigator commands             off
 errors                         off
 adhoc records                  off
 replication commands           off

(return status = 0)
1> sp_auditoption "enable auditing", "on"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
1> sp_auditoption
2> go
 name                           sval
 ------------------------------ ------------------------------
 logins                         off
 logouts                        off
 server boots                   off
 rpc connections                off
 roles                          off
 sa commands                    off
 sso commands                   off
 oper commands                  off
 navigator commands             off
 errors                         off
 adhoc records                  off
 replication commands           off
 enable auditing                on

(return status = 0)
```

5. Arrange for the audited events to occur, and check *sysaudits* to see if the data is being recorded.

```
godzilla% isql -Ustudent43 -Pstudent43
1> sp_who
2> go
 spid   status       loginame                        hostname   blk
        dbname                         cmd
 ------ ------------ ------------------------------- ----------
-----
        ------------------------------ ----------------
      1 running      student43                       godzilla   0
        master                         SELECT
```

```
      2 sleeping     NULL                                              0
        master                          NETWORK HANDLER
      3 sleeping     NULL                                              0
        master                          MIRROR HANDLER
      4 runnable     NULL                                              0
        sybsecurity                     AUDIT PROCESS
      5 sleeping     NULL                                              0
        master                          CHECKPOINT SLEEP

(5 rows affected, return status = 0)
1> use db42
2> go
```

Here are the corresponding entries in the sybsecurity..sysaudits table:

```
1> use sybsecurity
2> go
1> select * from sysaudits
2> go
 event  eventmod spid   eventtime                 sequence suid
dbid
         objid       xactid         loginname
         dbname                         objname
         objowner
         extrainfo


 ------ -------- ------ -------------------------- -------- ------
------
         ----------- -------------- ------------------------------
         ------------------------------
------------------------------
         ------------------------------

------------------------------------------------------------------
------
------------------------------------------------------------------
--------------
------------------------------------------------------------------
--------------
-----------------------
     1        1      1        Nov  3 1993 10:10AM        1      3
NULL
                NULL 0x0000058c000f student42
         NULL                              NULL
         NULL
         NULL


   107       1      1       Nov  3 1993 10:12AM        1      8
1
                NULL NULL            student43
         master                            NULL
         NULL
         goreset
sp_who
```

```
    107       1      1       Nov  3 1993 10:12AM       1      8
1
                NULL NULL           student43
          master                        NULL
          NULL
          sp_who


    107       1      1       Nov  3 1993 10:12AM       1      8
1
                NULL NULL           student43
          master                        NULL
          NULL
          use db42


    100       1      1       Nov  3 1993 10:12AM       1      8
7
                NULL NULL           student43
          db42                          NULL
          NULL
          USE


    107       1      1       Nov  3 1993 10:13AM       1      8
7
                NULL NULL           student43
          db42                          NULL
          NULL
          use sybsecurity


    107       1      1       Nov  3 1993 10:13AM       1      8
5
                NULL NULL           student43
          sybsecurity                   NULL
          NULL
          select * from sysaudits


(7 rows affected)
```

For readability, we'll delete all the rows from sysaudits before the next few commands.

```
godzilla% isql -Ustudent42 -Pstudent42
1> use sybsecurity
2> go
1> truncate table sysaudits
2> go
1> exec db42..sso_only
2> go
```

```
Success!
(return status = 0)
1> insert db42..sm_table values (1, "I'm being watched....")
2> go
(1 row affected)
```

And, the corresponding rows from sybsecurity..sysaudits:

```
1> select * from sybsecurity..sysaudits
2> go
 event  eventmod spid   eventtime                  sequence suid
dbid
         objid       xactid         loginname
         dbname                         objname
         objowner
         extrainfo


 ------ -------- ------ -------------------------- -------- ------
------
        ----------- ------------- ------------------------------
        ------------------------------
------------------------------
        ------------------------------

------------------------------------------------------------------
------
------------------------------------------------------------------
--------------
------------------------------------------------------------------
--------------
-----------------------
   103       1     1       Nov  3 1993 10:15AM       1     3
7
           48003202 NULL           student42
         db42                           sso_only
         NULL
         NULL



   101       1     1       Nov  3 1993 10:15AM       1     3
7
          128003487 NULL           student42
         db42                           sm_table
         dbo
         INSERT



(2 rows affected)
```

6. Write the full commands you would use to enable auditing of:

a) the execution of sp_addlogin

```
sp_auditsproc sp_addlogin, sybsystemprocs, "both"
```

b) all failed login attempts.

```
sp_auditoption "logins", "fail"
```

Lab 12a - Practicing System Administration Skills

1. **The first order of priority is to plan. Write down the tasks you will need to perform in order to successfully migrate the database? Keep in mind that all the data you have is available from the current SQL Server install with your database.**

---

   a. **Gather and record information about the database being moved in order to recreate it correctly in the new environment.**
   b. **Verify the integrity of the database using dbcc.**
   c. **Dump the database then verify the dump.**
   d. **Install the SQL Server, PROD2 (instructor demonstration).**
   e. **Reconfigure Server to handle the defined number of devices, users, and open database, 90, 100, and 25 respectively. (instructor demonstration)**
   f. **Shutdown and restart the server to activate the new configuration options. (instructor demonstration)**
   g. **Add the login and roles for the login of the user who is dbo of the deptNN database. Grant dbo create database permission.**
   h. **Log in as dbo, then create the appropriate devices that will be used by database deptNN. Note: Be sure to use your student number, for example pdata_dev99, in the name of the devices.**
   i. **Create the database, deptNN.**
   j. **Load the backup of the database into the newly created database and confirm successful loading of the database into the appropriate devices.**
   k. **Set up the remaining logins, users, and permissions.**
   l. **Drop the database dbNN and its devices from the previous SQL Server installation.**

   Note: The exercises that follow will use db42 and student42 as the example database and dbo respectively in the case study that follows. You will substitute your student identification, for example, 55, in place of "42" for the exercises that follow.

2. **Check your plan against the plan in the answer key. Make any appropriate changes if required. Write out the commands you would write for each task in the plan in the space provided below. Complete writing out the information before actually implementing the plan.**
   Note: Plan steps are those indicated in the answer key. If you wish to revise based on your plan, please do so.
   a. **Gather and record information about the database being moved in order to re-create it correctly in the new environment.**

First gather information about db42 from master, in order to recreate it correctly in the new environment:

nr. 12

```
sp_helpdb db42
go
```

Check sysusages to find out how the database was originally created and how it was subsequently altered. After checking sysusages, look at syssegments to confirm the existence of any user-defined segments:

```
select * from sysusages where dbid = <db_id_number>
go
use db42
go
select * from syssegments
go
```

Gather information about the dbo of the database to recreate the login with its associated roles on the new server. Since you are already logged in as the owner:

```
sp_displaylogin
go
```

b. Verify the integrity of the database using dbcc.

Since db42 is the current database for you, the dbcc commands for checking database integrity do not require the database name as arguments. They are included for completeness:

```
dbcc checkdb(db42)
go
dbcc checkalloc(db42)
go
dbcc checkcatalog(db42)
go
```

c. Dump the database, then verify the dump.

For your purposes, dump the database to a disk file named db42_dump in the current directory:

```
dump database db42 to "db42_dump"
go
```

Upon completing the dump, verify the dump using the "with headeronly" option of the load database command:

```
load database db42 from "db42_dump" with headeronly
go
```

d. Install the SQL Server, PROD2. This task needs to be an instructor demonstration.

e. Reconfigure Server to handle the defined number of devices, users, and open databases—90, 100, and 25 respectively. This task needs to be an instructor demonstration.

To set up the appropriate environment on the new server for this database (and other planned databases), restarting of the server has to be done while logged in as sybase. First show the current configuration then configure each of the options, then run the reconfigure option and confirm.

```
sp_configure
go
sp_configure "open databases", 25
go
sp_configure "devices", 90
go
sp_configure "user connections", 100
go
reconfigure
go
sp_configure
go
```

f. Shutdown and restart the Server to activate the new configuration options. This task needs to be an instructor demonstration.

```
shutdown
go
```

At the OS prompt type:

```
startserver -f RUN_PROD2
```

g. Add the login and roles for the login of the user who is dbo of the dept42 database. Grant dbo create database permission.

First log in to the server as sa:

```
isql -Usa -P -SPROD2
```

Set up the dbo, student42:

```
sp_addlogin student42, student42, master
go
sp_role "grant", "sa_role", student42
go
sp_role "grant", "sso_role", student42
go
sp_adduser student42
go
```

```
sp_displaylogin student42 -- confirm login setup correctly
go
grant create database to student42
go
```

h. Log in as dbo, then create the appropriate devices that will be used by database dept42. Note: Use the same device numbers as you did earlier in the class (see Lab 3a).

First, exit to the OS, then log in as student42:

```
isql -Ustudent42 -Pstudent42 -SPROD2
```

For this task we will use UNIX files, created in the current directory, as follows:

Name	Physical Name	Size (2k pages)	vdevno
pdata_dev42	pdata_dev42.dat	2048	2
plog_dev42	plog_dev42.dat	1024	3
pindex_dev42	pindex_dev42.dat	1024	4

The commands are:

```
disk init
name = pdata_dev42,
physname = "pdata_dev42.dat",
size = 2048,
vdevno = 2
go
disk init
name = plog_dev42,
physname = "plog_dev42.dat",
size = 1024,
vdevno = 3
go
disk init
name = pindex_dev42,
physname = "pindex_dev42.dat",
size = 1024,
vdevno = 4
go
```

i. Create the database dept42 according to the information gathered in 2.a.

Since you are going to be loading this database from a dump, use the "for load" option:

```
create database dept42
on pdata_dev42 = 2
log on plog_dev42 = 1 for load
go
```

```
alter database dept42 on pindex_dev42 = 1 for load
go
```

Because the device layout of the new database precisely matches that of the one you are about to load, you do not need to reconstruct segment information in advance; load database will take care of modifying the sysusages entries appropriately. In fact, having created the database "for load", there is no way to recreate that information!

Check sysusages and see the "before" image (image before loading database). This serves as a point of reference for verifying an accurate database migration:

```
select * from sysusages where dbid = db_id("dept42")
go
```

j. Load the backup of the database into the newly created database and confirm successful loading of the database into the appropriate devices.

```
load database dept42 from "db42_dump"
go
```

Verify that dept42 database was correctly loaded:

```
select * from sysusages where dbid = db_id("dept42")
go
sp_helpdb dept42
go
```

Note: Locking The Database
If after loading the database you had transaction logs to load, you would need to take steps to prevent users from accessing the database during that recovery effort. If users perform logged transactions between loading the database and the first log dump, or between the loading of two transaction log dumps, the subsequent log load will fail. The database can be locked before you load it to ensure that users do not make changes. To achieve this, use sp_dboption to set the 'no chkpt on recovery', 'dbo use only', and 'read only' options to TRUE. Do not reset these options until the last transaction log has been loaded.

k. Set up the remaining logins, users, and permissions. (This is a paper-only activity)

Note: Ideally, suids need to match between users (in the database) and logins (in the server). Because there has been no set order to the addition of logins to this server during the class, it will not be practical to perform this exercise in the classroom environment. It does, however, highlight the desirability of maintaining scripts for all such server activities!

1. Gather information about each user in the db42 database using sp_helpuser.

2. Gather information about each user's login information using sp_displaylogin or you could retrieve the information directly from the appropriate system tables (syslogins, sysroles, sysloginroles).

3. Add these logins to the new server in the correct order using sp_addlogin. This ensures that the logins will receive the same suid.

4. Reconfigure the roles using sp_role.

l. Drop the database db42 and its devices from the previous SQL Server installation. First exit the new Server installation, enter the older installation, then:

```
drop database db42
go
sp_dropdevice data_dev42
go
sp_dropdevice log_dev42
go
sp_dropdevice index_dev42
go
drop database db42
go
sp_dropdevice data_dev42
go
sp_dropdevice log_dev42
go
sp_dropdevice index_dev42
go
```

3. Execute the above tasks on the SQL Server. The output for each of the above tasks will be shown below:

a. Gather and record information about the database being moved in order to re-create it correctly in the new environment.

First gather information about db42 from master, in order to recreate it correctly in the new environment:

```
1> sp_helpdb db42
2> go
 name                 db_size       owner                 dbid
        created
        status

 ---------------------- ------------ ---------------------- ------
       --------------

------------------------------------------------------------------------
------------------------------
 db42                          4.0 MB student42                    7
        Nov 01, 1993
```

```
        select into/bulkcopy


 device_fragments               size          usage                free kbytes
 ------------------------------ ------------- -------------------- -----------
 data_dev42                     2.0 MB        data only                   1296
 index_dev42                    1.0 MB        data only                    976
 log_dev42                      1.0 MB        log only                     640

(return status = 0)
```

This shows that the database is spread out over three devices.

Check sysusages to find out how the database (dbid = 7) was originally created and how it was subsequently altered. After checking sysusages, look at syssegments to confirm the existence of any user-defined segments:

```
1> select * from sysusages where dbid = 7
2> go
 dbid   segmap      lstart      size        vstart      pad    unreservedpgs
 ------ ----------- ----------- ----------- ----------- ------ -------------
      7           3           0        1024    50331648   NULL           648
      7           4        1024         512    67108864   NULL           320
      7           8        1536         512    83886080   NULL           488

(3 rows affected)
```

From this we can glean that the database was originally created with data (2Mb) and log (1Mb) on separate devices. It was subsequently altered onto a third device (1Mb). This third fragment then had a user-defined segment added to it, and system and default dropped from it, hence the segmap value of 8. Looking at syssegments in db42 will confirm the existence of the user-defined segment:

```
1> use db42
2> go
1> select * from syssegments
2> go
 segment name                           status
 ------- ------------------------------ ------
       0 system                              0
       1 default                             1
       2 logsegment                          0
       3 seg1                                0

(4 rows affected)
```

Gather information about the dbo of the database to recreate the login with its associated roles on the new server. Since you are already logged in as the owner:

```
1> sp_displaylogin
2> go
Suid: 3
Loginame: student42
Fullname:
Configured Authorization: sa_role sso_role
Locked: NO
Date of Last Password Change: Nov  1 1993  2:45PM
(return status = 0)
```

We will need to grant the sa and sso roles to student42 in the new server.

b. Verify the integrity of the database using dbcc.

Since db42 is the current database, the dbcc commands for checking database integrity do not require the database name as arguments. They are included for completeness:

```
1> dbcc checkdb(db42)
2> go
Checking db42
Checking 1
The total number of data pages in this table is 2.
Table has 29 data rows.
Checking 2
The total number of data pages in this table is 3.
Table has 33 data rows.
<etc. Large output omitted>

1> dbcc checkalloc(db42)
2> go
Database 'db42' is not in single user mode - may find spurious allocation
problems due to transactions in progress.
************************************************************
TABLE: sysobjects               OBJID = 1
INDID=1  FIRST=1         ROOT=8  SORT=0
        Data level: 1.  2 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=16        ROOT=16         SORT=0
        Indid     : 2.  1 Index Pages in 1 extents.
TOTAL # of extents = 3
************************************************************
TABLE: sysindexes               OBJID = 2
INDID=1  FIRST=24        ROOT=32         SORT=0
        Data level: 1.  3 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
TOTAL # of extents = 2
************************************************************
<etc. Large output omitted>
```

```
1> dbcc checkcatalog(db42)
2> go
Checking db42
The following segments have been defined for database 7 (database name db42).
virtual start addr      size      segments
--------------------    ------    -------------------------
50331648                      1024
                                     0
                                     1
67108864                      512
                                     2
83886080                      512
                                     3
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

Hopefully, the output from the dbcc commands indicates that there are no problems. (If there had been, your next course of action would be to refer to the System Administration Guide, refer to the Troubleshooting Guide, and/or call Sybase Technical Support for advice on how to proceed.) Having verified the database, proceed to dump it to an appropriate device.

c. Dump the database, then verify the dump.

For your purposes, dump the database to a disk file named db42_dump in the current directory:

```
1> dump database db42 to "db42_dump"
2> go
Backup Server session id is:  21.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db4294042073EE   ' section number 0001
mounted on disk file '/curdev1/server10ga/install/db42_dump'
Backup Server: 4.58.1.1: Database db42: 642 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database db42: 650 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database db42).
```

Upon completing the dump, verify the dump using the "with headeronly" option of the load database command:

```
1> load database db42 from "db42_dump" with headeronly
2> go
Backup Server session id is:  24.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db4294042073EE   ' section number 0001
mounted on disk file '/curdev1/server10ga/install/db42_dump'
This is a database dump of database ID 7 NAME db42 from Feb 11 1994  8:14AM.
```

```
SQL Server Version: SQL Server/10.0/P/Sun4/OS 4.1.x/1/OPT/Wed Sep 22 12:52:03
PDT 1993.
Database contains 2048 pages; checkpoint RID=(Rid pageid = 0x5a6; row num =
0x17); next object ID=384004399; sort order ID=50, status=0; charset ID=1.
```

d. Install the SQL Server, PROD2. This task needs to be an instructor demonstration (you must be logged in as sybase for installing, reconfiguring, shutting down and restarting the SQL Server).

e. Reconfigure Server to handle the defined number of devices, users, and open databases—90, 100, and 25 respectively. This task needs to be an instructor demonstration.

To set up the appropriate environment on the new server for this database (and other planned databases), restarting of the server has to be done while logged in as sybase. First show the current configuration then configure each of the options, then run the reconfigure option and confirm.

```
1> sp_configure
2> go
<very large output omitted>

1> sp_configure "open databases", 25
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> sp_configure "devices", 90
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> sp_configure "user connections", 100
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> reconfigure
2> go
```

Note: It is not strictly necessary to run the RECONFIGURE command after each sp_configure; if they are being done consecutively, as here, one RECONFIGURE would suffice.

```
1> sp_configure -- to confirm new configuration settings
2> go
<very large output omitted>
```

f. Shutdown and restart the Server to activate the new configuration options.

```
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
```

At the OS prompt type:

```
startserver -f RUN_PROD2

00:94/02/11 09:08:27.40 kernel  Using config area of disk for boot information
00:94/02/11 09:08:27.51 kernel  Using config area from primary master device.
00:94/02/11 09:08:27.71 kernel  SQL Server/10.0/P/Sun4/OS 4.1.x/1/OPT/Wed Sep
22 12:52:03 PDT 1993
00:94/02/11 09:08:27.71 kernel  Confidential Property of Sybase, Inc.
00:94/02/11 09:08:27.71 kernel  Copyright Sybase, Inc. 1987, 1993.
00:94/02/11 09:08:27.71 kernel  All rights reserved.
00:94/02/11 09:08:27.71 kernel  Use, duplication, or disclosure by the United
States Government
00:94/02/11 09:08:27.72 kernel  is subject to restrictions as set forth in FARS
subparagraphs
00:94/02/11 09:08:27.72 kernel  52.227-19(a)-(d) for civilian agency contracts
and DFARS
00:94/02/11 09:08:27.72 kernel  252.227-7013(c)(1)(ii) for Department of
Defense contracts.
00:94/02/11 09:08:27.73 kernel  Sybase reserves all unpublished rights under
the copyright
00:94/02/11 09:08:27.74 kernel  laws of the United States.
00:94/02/11 09:08:27.74 kernel  Sybase, Inc. 6475 Christie Avenue, Emeryville,
CA 94608, USA.
00:94/02/11 09:08:27.74 kernel  Logging SQL Server messages in file
'/curdev1/server10ga/install/PROD2_errorlog'
.
00:94/02/11 09:08:27.80 kernel  Using 256 file descriptors.
00:94/02/11 09:08:27.80 kernel  Network and device connection limit is 253.
00:94/02/11 09:08:28.09 kernel  Initializing virtual device 0,
'/curdev1/server10ga/devices/PROD2_master.dat'
00:94/02/11 09:08:28.09 kernel  Virtual device 0 started using standard unix
i/o
.
00:94/02/11 09:08:28.10 server  Not enough memory for initialization -
requested
```

```
 3511956
00:94/02/11 09:08:28.11 server  Physical memory on this machine may be too
fragmented
00:94/02/11 09:08:28.11 kernel  ueshutdown: exiting
```

Clearly, the server has failed to start. The memory requirements of the configuration changes were not taken into account, and you need to recover from this condition. This can be done using buildmaster, with the '-r' option, which rewrites the configuration parameters to their default values. You will need the location of the master device, from the RUN_PROD2 runserver file. You will then need to:

1. **Restart the Server.**
2. **Calculate the memory requirements (see below)**

	Default Value	New Value	Diff	Mem Per	Total Kb	# 2k Pages
devices	10	90	80	.5	40	20
users	25	100	75	50	3750	1875
databases	10	25	15	7	175	88

						1983

This shows that we need to add 1983 (round this value to 2000) pages of memory to the default value of 5120, giving a final value of 7120.

3. **Reconfigure the server including the memory.**
4. **Shutdown then restart the Server**

The commands to recover and reconfigure the Server are as follows:

```
$SYBASE/bin/buildmaster -d/curdev1/server10ga/devices/PROD2_master.dat  -r
Master device: /curdev1/server10ga/devices/PROD2_master.dat
reading configuration area
writing configuration area
Buildmaster complete

startserver -f RUN_PROD2
<startserver output omitted>

isql -Usa -P -SPROD2
1> sp_configure "memory", 7120
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> sp_configure "devices", 90
2> go
Configuration option changed. Run the RECONFIGURE command to install.
```

```
(return status = 0)
1> sp_configure "user connections", 100
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> sp_configure "open databases", 25
2> go
Configuration option changed. Run the RECONFIGURE command to install.
(return status = 0)
1> reconfigure
2> go
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
00:94/02/11 09:28:23.07 server  SQL Server shutdown by request.
00:94/02/11 09:28:23.10 kernel  ueshutdown: exiting
DB-LIBRARY error:
        Unexpected EOF from SQL Server.


startserver -f RUN_PROD2
<startserver output omitted>
```

g. Add the login and roles for the login of the user who is dbo of the dept42 database. Grant dbo create database permission.

First log in to the server from the OS:

```
isql -Usa -P -SPROD2
```

Set up the dbo, student42:

```
1> sp_addlogin student42, student42, master
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
1> sp_role "grant", "sa_role", student42
2> go
Authorization updated.
(return status = 0)
1> sp_role "grant", "sso_role", student42
2> go
Authorization updated.
(return status = 0)
1> sp_displaylogin student42
```

```
2> go
Suid: 3
Loginame: student42
Fullname:
Configured Authorization: sa_role sso_role
Locked: NO
Date of Last Password Change: Feb 11 1994  9:33AM
(return status = 0)
1> sp_adduser student42
2> go
New user added.
(return status = 0)
1> grant create database to student42
2> go
```

h. Log in as dbo, then create the appropriate devices that will be used by database dept42. Note: Use the same device numbers as you did earlier in the class (see Lab 3a).

First, exit to the OS, then log in as student42:

```
isql -Ustudent42 -Pstudent42 -SPROD2
```

For this task we will use UNIX files, created in the current directory, as follows:

Name	Physical Name	Size (2k pages)	vdevno
pdata_dev42	pdata_dev42.dat	2048	2
plog_dev42	plog_dev42.dat	1024	3
pindex_dev42	pindex_dev42.dat	1024	4

```
1> disk init
2> name = pdata_dev42,
3> physname = "pdata_dev42.dat",
4> size = 2048,
5> vdevno = 2
6> go
1> disk init
2> name = plog_dev42,
3> physname = "plog_dev42.dat",
4> size = 1024,
5> vdevno = 3
6> go
1> disk init
2> name = pindex_dev42,
3> physname = "pindex_dev42.dat",
4> size = 1024,
5> vdevno = 4
6> go
```

i. Create the database dept42.

Since you are going to be loading this database from a dump, use the "for load" option:

```
1> create database dept42
2> on pdata_dev42 = 2
3> log on plog_dev42 = 1 for load
4> go
CREATE DATABASE: allocating 1024 pages on disk 'pdata_dev42'
CREATE DATABASE: allocating 512 pages on disk 'plog_dev42'

1> alter database dept42 on pindex_dev42 = 1 for load
2> go
Extending database by 512 pages on disk pindex_dev42
```

Note what happens if we try to use the database before loading it:

```
1> use dept42
2> go
Msg 930, Level 14, State 1:
Line 1:
Database 'dept42' cannot be opened because either an earlier system
termination
left LOAD DATABASE incomplete or the database is created with 'for load'
option.
Load the database or contact a user with System Administrator (SA) role.
```

Because the device layout of the new database precisely matches that of the one we are about to load, we do not need to reconstruct segment information in advance; load database command will take care of modifying the sysusages entries appropriately. In fact, having created the database "for load", there is no way to recreate that information!

Check sysusages and see the "before" image (image before loading database):

```
1> select * from sysusages where dbid = db_id("dept42")
2> go
 dbid   segmap      lstart      size        vstart      pad    unreservedpgs
 ------ ----------- ----------- ----------- ----------- ------ -------------
      5           3           0        1024    33554432   NULL           688
      5           4        1024         512    50331648   NULL           504
      5           3        1536         512    67108864   NULL           512

(3 rows affected)
```

j. Load the backup of the database into the newly created database and confirm successful loading of the database into the appropriate devices.

```
1> load database dept42 from "db42_dump"
2> go
Backup Server session id is:  15.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'db4294042073EE   ' section number
0001
mounted on disk file '/curdev1/server10ga/install/db42_dump'
Backup Server: 4.58.1.1: Database dept42: 4100 kilobytes LOADed.
Backup Server: 4.58.1.1: Database dept42: 4108 kilobytes LOADed.
00:94/02/11 10:01:30.36 server  Recovery dbid 5 ckpt (1446,23)
00:94/02/11 10:01:30.44 server  Recovery no active transactions before ckpt.
Backup Server: 3.42.1.1: LOAD is complete (database dept42).
```

Verify that dept42 database was correctly loaded:

```
1> select * from sysusages where dbid = db_id("dept42")
2> go
 dbid   segmap      lstart      size        vstart      pad    unreservedpgs
 ------ ----------- ----------- ----------- ----------- ------ -------------
      5           3           0        1024    33554432   NULL           648
      5           4        1024         512    50331648   NULL           320
      5           8        1536         512    67108864   NULL           488

(3 rows affected)

sp_helpdb dept42
go
 name                    db_size       owner                    dbid
        created
        status

 ----------------------- ------------ ---------------------- ------
       --------------

------------------------------------------------------------------------
------------------------------
 dept42                          4.0 MB student42                    5
        Feb 11, 1994
        no options set


 device_fragments               size          usage                free
kbytes
 ------------------------------ ------------- --------------------
```

```
-----------
 pdata_dev42                    2.0 MB        data only
1296
 pindex_dev42                   1.0 MB        data only
 976
 plog_dev42                     1.0 MB        log only
 640

(return status = 0)
```

Note: Locking The Database

If after loading the database you had transaction logs to load, you would need to take steps to prevent users from accessing the database during that recovery effort. If users perform logged transactions between loading the database and the first log dump, or between the loading of two transaction log dumps, the subsequent log load will fail. The database can be locked before you load it to ensure that users do not make changes. To achieve this, use sp_dboption to set the 'no chkpt on recovery', 'dbo use only', and 'read only' options to TRUE. Do not reset these options until the last transaction log has been loaded.

k. Set up the remaining logins, users, and permissions.

Note: In the real world, suids need to match between users (in the database) and logins (in the server). Because there has been no set order to the addition of logins to this server during the class, it will not be practical to perform this exercise in the classroom environment. It does, however, highlight the desirability of maintaining scripts for all such server activities!

The following is for illustration purposes only.

1. Gather information about each user in the db42 database using sp_helpuser.

```
1> use db42
2> go
1> sp_helpuser
2> go
 Users_name         ID_in_db Group_name         Login_name         Default_db
 ------------------ -------- ------------------ ------------------ --------------
 dbo                       1 public             sa                 master
 dick42                    4 programmers        dick42             db42
 harry42                   5 programmers        harry42            db42
 laura42                   6 public             laura42            db42
 tom42                     3 programmers        tom42              db42

(return status = 0)
```

2. Gather information about each user's login information using sp_displaylogin or you could retrieve the information directly from the appropriate system tables (syslogins, sysroles, sysloginroles).

```
1> sp_displaylogin dick42
2> go
Suid: 5
Loginame: dick42
Fullname:
Configured Authorization: sso_role
Locked: NO
Date of Last Password Change: Nov  2 1993 2:24PM
(return status = 0)
1> sp_displaylogin harry42
2> go
Suid: 6
Loginame: harry42
Fullname:
Configured Authorization:
Locked: NO
Date of Last Password Change: Nov 2 1993  2:25PM
(return status = 0)
1> sp_displaylogin laura42
2> go
Suid: 7
Loginame: laura42
Fullname:
Configured Authorization:
Locked: NO
Date of Last Password Change: Nov  2 1993  3:19PM
(return status = 0)
1> sp_displaylogin tom42
2> go
Suid: 4
Loginame: tom42
Fullname:
Configured Authorization: sa_role oper_role
Locked: NO
Date of Last Password Change: Nov  2 1993  2:24PM
(return status = 0)
```

3. **Add these logins to the new server in the correct order using sp_addlogin. This ensures that the logins will receive the same suid. Summarizing, we have:**

Loginame	Suid	Roles
tom42	4	sa_role oper_role
dick42	5	sso_role
harry42	6	(none)
laura42	7	(none)

```
1> sp_addlogin tom42, friday, dept42
2> exec sp_addlogin dick42, friday, dept42
3> exec sp_addlogin harry42, friday, dept42
```

```
4> exec sp_addlogin laura42, friday, dept42
5> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
```

4. Reconfigure the roles using sp_role.

```
1> sp_role "grant", "sa_role", tom42
2> go
Authorization updated.
(return status = 0)
1> sp_role "grant", "oper_role", tom42
2> go
Authorization updated.
(return status = 0)
1> sp_role "grant", "sso_role", dick42
2> go
Authorization updated.
(return status = 0)
```

l. Drop the database db42 and its devices from the previous SQL Server installation. First exit the new Server installation, then:

```
isql -Ustudent42 -Pstudent42 -SMARION10
1> drop database db42
2> go
1> sp_dropdevice data_dev42
2> go
Device dropped.
(return status = 0)
1> sp_dropdevice log_dev42
2> go
Device dropped.
(return status = 0)
1> sp_dropdevice index_dev42
```

```
2> go
Device dropped.
(return status = 0)
```

Lab 12b - Mini-Case Studies: Solving Common Systems Administration Problems(optional)

The following are paper-based activities, with the exception of exercise 15. Write your answers in the spaces provided. You may wish to confirm some of the commands you propose as solutions to the scenarios on the SQL Server.

RECOVERABILITY AND MIRRORING

1. **You have moved a database from development to production. Requirements for the production database are:**
   - data and log are mirrored separately
   - database is recoverable from transaction log dumps
   Examine the output below.

```
1> sp_helpdb devel42
2> go
 name                 db_size       owner                 dbid
        created
        status

 -------------------- ------------ ---------------------- ------
       --------------

-------------------------------------------------------------------------
------------------------------
 devel42                 15.0 MB sa                           8
        Feb 15, 1994
        trunc log on chkpt

 device_fragments            size          usage               free kbytes
 --------------------------- ------------- ------------------- -----------
 devel_dev                   15.0 MB       data and log              14672
(return status = 0)
```

Is the database configured correctly to satisfy the production requirements? If not, what is the problem(s) and how will it be corrected?

To satisfy the production requirements:

1) the "trunc log on chkpt" option needs to be turned off.

2) the database needs a separate log segment; use alter database to extend the database onto a new device (disk init if necessary), and then use sp_logdevice to move the log to the new device.

3) the data and log may now be mirrored as required.

SEPARATING DATABASES ON A SHARED DEVICE

2. There are two databases, db1 and db2, on a single server. Both databases have their data segments on a device called data_dev1, and their logs on a device called log_dev1. To accommodate growth in the databases, you need to move db2's data segments to a new device. What steps do you take to achieve this?

---

a) disk init the new device, data_dev2, as appropriate.

b) dump database db2 to the device of your choice.

c) drop database db2

d) create database db2 on data_dev2 and log_dev1, sized as before

e) load db2 from the dump

f) now both db1 and db2 can be altered as required to allocate more space.

STANDARDIZING DATABASE CONFIGURATION

3. Your organization has standard procedures, rules, defaults, user-defined datatypes, and sp_dboptions that would be used for 90% of the databases that would be created in your organization. How would you streamline the incorporation of all these objects and options into future databases you create?

The simplest way of dealing with this would be to put all the common objects in model. However, you should be careful as far as space is concerned; if you need to increase the size of model, the server will not permit you to make it larger than tempdb. If model becomes bigger than tempdb, tempdb will have to be altered.

An alternate method for handling this would be to use scripts to recreate the common objects in the databases that require them. A third approach, sometimes used for stored procedures, is to put them all in their own database (thus emulating the Server's own sybsystemprocs).

WORKING WITH SEGMENTS

4. You have a large table in your database, dept42, that you want to split evenly across two devices (tab_dev1 and tab_dev2). You already have the ability to load the data in two portions. What procedure would you use? You will be using the following stored procedures: sp_addsegment, sp_dropsegment, sp_extendsegment.

---

1. First, alter the database to add the two new devices for the large table:

2. Show the new layout. Recall that the newly allocated fragments will have the system and default segments mapped to them automatically:

3. Drop system and default segments from the new devices, to prevent unintended allocations:

4. Add a new segment, tab_seg1, to the first of the new devices:

5. Create the table, and insert some data:

6. Extend tab_seg1 to the second device, tab_dev2:

7. Drop tab_seg1 from tab_dev1, to ensure no further allocation on that device:

8. Insert the second batch of rows:

9. Extend tab_seg1 back onto tab_dev1, to ensure that future allocation for this table can come from either device:

10. Finally, run sp_helpdb to verify the equal distribution of data. If you refer back to the earlier sp_helpdb, you will see that the free space on both tab_dev1 and tab_dev2 has been reduced by the same amount (was 4096Kb, is now 4032Kb).

(see the sample output below for each step in the procedure above)

1. First, alter the database to add the two new devices for the large table:

```
1> alter database dept42 on tab_dev1 = 4
2> go
Extending database by 2048 pages on disk tab_dev1
1> alter database dept42 on tab_dev2 = 4
2> go
Extending database by 2048 pages on disk tab_dev2
```

2. Show the new layout. Recall that the newly allocated fragments will have the system and default segments mapped to them automatically:

```
1> use dept42
2> go
1> sp_helpdb dept42
2> go
 name                          db_size        owner                         dbid
        created
        status

 ------------------------ ------------ ------------------------ ------
       --------------

------------------------------------------------------------------------
------------------------------
 dept42                         12.0 MB sa                              5
        Feb 15, 1994
        trunc log on chkpt


 device_fragments               size          usage                free kbytes
```

```
------------------------------ ------------ ------------------- -----------
pdata_dev42                    2.0 MB       data only                  1296
pindex_dev42                   1.0 MB       data only                   976
plog_dev42                     1.0 MB       log only                   1008
tab_dev1                       4.0 MB       data only                  4096
tab_dev2                       4.0 MB       data only                  4096

device                         segment
------------------------------ ------------------------------
pdata_dev42                    default
pdata_dev42                    system
pindex_dev42                   seg1
plog_dev42                     logsegment
tab_dev1                       default
tab_dev1                       system
tab_dev2                       default
tab_dev2                       system

(return status = 0)
```

3. Drop system and default segments from the new devices, to prevent unintended allocations:

```
1> sp_dropsegment "default", dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment reference to device dropped.
(return status = 0)
1> sp_dropsegment "default", dept42, tab_dev2
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment reference to device dropped.
(return status = 0)
1> sp_dropsegment system, dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment reference to device dropped.
WARNING: There are no longer any segments referencing device 'tab_dev1'.  This
device will no longer be used for space allocation.
(return status = 0)
1> sp_dropsegment system, dept42, tab_dev2
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment reference to device dropped.
WARNING: There are no longer any segments referencing devices 'tab_dev1,
```

```
tab_dev2'.  These devices will no longer be used for space allocation.
(return status = 0)
```

4. Add a new segment, tab_seg1, to the first of the new devices:

```
1> sp_addsegment tab_seg1, dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment created.


(return status = 0)
```

5. Create the table, and insert some data:

```
1> create table big_tab (i int, c char(255)) on tab_seg1
2> go
1> insert big_tab values (1, "dummy data") -- This is dummy data
2> go 200
(1 row affected)
200 xacts:
```

6. Extend tab_seg1 to the second device, tab_dev2:

```
1> sp_extendsegment tab_seg1, dept42, tab_dev2
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment extended.
(return status = 0)
```

7. Drop tab_seg1 from tab_dev1, to ensure no further allocation on that device:

```
1> sp_dropsegment tab_seg1, dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment reference to device dropped.
WARNING: There are no longer any segments referencing device 'tab_dev1'.  This
device will no longer be used for space allocation.
(return status = 0)
```

8. Insert the second batch of rows:

```
1> insert big_tab values(2, "more dummy data")
2> go 200
(1 row affected)
200 xacts:
```

9. Extend tab_seg1 back onto tab_dev1, to ensure that future allocation for this table can come from either device:

```
1> sp_extendsegment tab_seg1, dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment extended.
(return status = 0)
```

10. Finally, run sp_helpdb to verify equal distribution of data. If you refer back to the earlier sp_helpdb, you will see that the free space on both tab_dev1 and tab_dev2 has been reduced by the same amount (was 4096Kb, is now 4032Kb).

```
1> sp_helpdb dept42
2> go
 name                          db_size        owner                         dbid
         created
         status

 ------------------------ ------------ ------------------------ ------
        --------------

-------------------------------------------------------------------------------
------------------------------
 dept42                        12.0 MB sa                             5
         Feb 15, 1994
         trunc log on chkpt


 device_fragments              size          usage                free kbytes
 ----------------------------- ------------- -------------------- -----------
 pdata_dev42                   2.0 MB        data only                  1296
 pindex_dev42                  1.0 MB        data only                   976
 plog_dev42                    1.0 MB        log only                   1008
 tab_dev1                      4.0 MB        data only                  4032
 tab_dev2                      4.0 MB        data only                  4032

 device                        segment
 ----------------------------- ------------------------------
 pdata_dev42                   default
 pdata_dev42                   system
 pindex_dev42                  seg1
 plog_dev42                    logsegment
 tab_dev1                      tab_seg1
 tab_dev2                      tab_seg1

(return status = 0)
```

MONITORING THE ERROR LOG

5. The login with OPER role performs database and transaction dumps of the production database on a daily basis. As sa, you carefully monitor the error log routinely and find the following information. What should you do?

Error log excerpt:

```
00:94/02/15 09:49:53.67 server WARNING: **************************
00:94/02/15 09:49:53.81 server Attempt by user 1 to dump xact on db db42
with NO_LOG
00:94/02/15 09:49:54.86 server Attempt by user 1 to dump xact on db db42
with NO_LOG was successful
00:94/02/15 09:49:54.86 server WARNING: **************************
```

Dump the database as soon as possible since a dump transaction with NO_LOG prevents a database recovery from a transaction dump. Advise the login with OPER role, that performed this transaction dump with NO_LOG, of this error.

6. Users are receiving the following messages at the client application. From the OS prompt, "%", the client sees:

```
% isql -Usa -P -SMARION10
DB-LIBRARY error:
      Unexpected EOF from SQL Server.
%
```

You check the error log and find the following:

```
00:94/02/15 09:52:54.50 server Error: 21, Severity: 21, State: 1
00:94/02/15 09:52:54.53 server WARNING - Fatal Error 1601 occurred at Feb
15 1994 9:52AM. Please note the error and time, and contact a user with
System Administrator (SA) authorization.
00:94/02/15 09:52:54.56 server Error: 1601, Severity: 21, State: 3
00:94/02/15 09:52:54.56 server There are not enough 'user connections'
available to start a new process. Retry when there are fewer active users,
or ask your System Administrator to reconfigure SQL Server with more user
connections.
```

What can you do to prevent this from happening again?

The SA will need to use sp_configure to increase the number of user connections, and possibly increase the amount of memory allocated to the server to compensate.

MONITORING SPACE USAGE

7. You have your database and some tables distributed on various devices and segments. One of the users complains that he/she is receiving the following message after attempting an insert on table, big_tab:

```
Msg 1105, Level 17, State 1:
Line 1:
Can't allocate space for object 'big_tab' in database 'dept42' because the
'tab_seg1' segment is full. If you ran out of space in syslogs, dump the
transaction log. Otherwise, use ALTER DATABASE or sp_extendsegment to
increase the size of the segment.
```

What could be the cause of this error and where would you look to find additional information to validate your target root cause?

The device on which big_tab table resides may no longer have any free space left. To verify:

1. Look at sp_help for the table.
2. Look at sp_helpdb for the database to find out about the segment on which big_tab resides.
3. Confirm that no free space resides on the device based on the output of sp_helpdb. If there is no free space, you may resolve the problem by extending the segment if appropriate.

8. For the solution to question 7, which commands/stored procedures would you execute to find the information and correct the problem?

The output of the above sequence of events in scenario 7 may be as follows:

```
1> sp_help big_tab
2> go
 Name                           Owner
         Type
 ------------------------------ ------------------------------
        ----------------------
 big_tab                        dbo
         user table

 Data_located_on_segment        When_created
 ------------------------------ --------------------------
 tab_seg1                              Feb 16 1994  2:06PM

 Column_name     Type             Length Prec Scale Nulls Default_name
         Rule_name       Identity
 --------------- ---------------- ------ ---- ----- ----- ---------------
        --------------- --------
 c1              char                255 NULL  NULL     0 NULL
         NULL                    0
```

```
 c2              char                255 NULL  NULL      0 NULL
        NULL                    0
 c3              char                255 NULL  NULL      0 NULL
        NULL                    0
 c4              char                255 NULL  NULL      0 NULL
        NULL                    0
Object does not have any indexes.
No defined keys for this object.

(return status = 0)
```

This shows the data to be located on a segment called tab_seg1. Now look at sp_helpdb for the database, to find out about tab_seg1:

```
1> sp_helpdb dept42
2> go
 name                    db_size       owner                    dbid
        created
        status

 ----------------------- ------------ ------------------------ ------
       --------------

-------------------------------------------------------------------------
------------------------------
 dept42                          12.0 MB sa                            5
        Feb 15, 1994
        trunc log on chkpt

 device_fragments               size          usage                free kbytes
 ------------------------------ ------------- -------------------- -----------
 pdata_dev42                    2.0 MB        data only                   1296
 pindex_dev42                   1.0 MB        data only                    976
 plog_dev42                     1.0 MB        log only                    1008
 tab_dev1                       4.0 MB        data only                   4096
 tab_dev2                       4.0 MB        data only                      0

 device                         segment
 ------------------------------ ------------------------------
 pdata_dev42                    default
 pdata_dev42                    system
 pindex_dev42                   seg1
 plog_dev42                     logsegment
 tab_dev1                        -- unused by any segments --
 tab_dev2                       tab_seg1

(return status = 0)
```

This shows that tab_seg1 is mapped to the device tab_dev2, and as indicated by the original error message, there is no free space left on this device. Note also that the fragment on tab_dev1 currently has no segments mapped to it, and is thus completely unused (This is a technique which can be used to reserve space on a device for future use by a database). The remedy here will be to extend the segment to tab_dev1. For example:

```
1> sp_extendsegment tab_seg1, dept42, tab_dev1
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Segment extended.
(return status = 0)
```

After performing an insert in big_tab, running sp_helpdb on the database will show space being used on tab_dev1.

THRESHOLDS

9. You have the following sp_thresholdaction procedure defined in sybsystemprocs:

```
create procedure sp_thresholdaction
    @dbname                 varchar(30),
    @segmentname            varchar(30),
    @space_left             int,
    @status                 int
as
declare @devname varchar(100),
        @before_size int,
        @after_size int,
        @before_time datetime,
        @after_time datetime,
        @error int

if (@status&1) = 1
begin
    print "LOG FULL: database '%1!'", @dbname
end

if @segmentname = (select name from syssegments
            where segment = 2)
begin
    select  @before_time = getdate(),
        @before_size = reserved_pgs(id, doampg)
    from sysindexes
    where sysindexes.name = "syslogs"

    print "LOG DUMP: database '%1!', threshold '%2!'",
```

```
        @dbname, @space_left

    select @devname = "/backup/" + @dbname + "_" +
        convert(char(8), getdate(),4) + "_" +
        convert(char(8), getdate(), 8)

    dump transaction @dbname to @devname
        /* error checking */
        select @error = @@error
        if @error != 0
        begin
            print "LOG DUMP ERROR: %1!", @error
        end

    /* get size of log and time after dump */
        select @after_time = getdate(),
            @after_size = reserved_pgs(id, doampg)
            from sysindexes
            where sysindexes.name = "syslogs"

        print "LOG DUMPED TO: device '%1!", @devname
        print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
            @before_size, @after_size
        print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time
    end

    else

    begin /* continued on next page */
        print "THRESHOLD WARNING: database '%1!', segment '%2!' at
    '%3!' pages", @dbname, @segmentname, @space_left
    end
```

a. What message is printed if the procedure is called as the result of reaching the log's last-chance threshold?

```
"LOG FULL: database '%1!'", @dbname
```

b. What is the segment ID for the transaction log segment?

The log segment is always segment id 2.

c. What information is written to the error log if the log segment is the threshold segment?

```
print "LOG DUMP: database '%1!', threshold '%2!'",
                @dbname, @space_left
print "LOG DUMPED TO: device '%1!", @devname
        print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
```

```
            @before_size, @after_size
        print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time
```

d. How could the system be set up so that all segment thresholds point to the sp_thresholdaction procedure in sybsystemprocs?

Define all thresholds to run the sp_thresholdaction procedure, and make sure that the only such procedure on the server is in sybsystemprocs.

e. What information is written to the error log if the threshold segment is not the log segment?

```
"THRESHOLD WARNING: database '%1!', segment '%2!' at '%3!' pages", @dbname,
@segmentname, @space_left
```

f. What would you change in the procedure to dump the transaction to a tape device named "/dev/nrst1"

Change the line that reads:

```
dump transaction @dbname to @devname
```

to:

```
dump transaction @dbname to "/dev/nrst1"
```

This will cause the dump to be appended to those already on the tape.

g. If the log does not shrink much after the dump transaction, what could be happening?

If the log does not shrink much after the dump, the most likely cause is that there is a long-running transaction taking place in the database. Because dump transaction can only clear out the log up to the beginning of the oldest active transaction, it is possible for a long-running transaction to prevent any further truncation until it completes.

10. You want to override the above sp_thresholdaction procedure for your database dbNN. The above procedure cannot be changed. You want to set the threshold for the data segment of that database at 200 pages. The threshold procedure then prints an error message to the error log. Answer the following questions:

a. Could you create a procedure of the same name, sp_thresholdaction, in the database? Why or why not?

You would have to call the procedure something different. If the scenario in the previous question is in place, the Last Chance Threshold in this database is already associated with sp_thresholdaction, and it is intended to be the one in sybsystemprocs; providing a similarly named procedure local to the database would override the procedure in sybsystemprocs.

b. What procedure would you follow to set this threshold procedure up for your database?

Define the threshold in dbNN as follows. Assume your new procedure is to be called sp_dataspace:

```
1> sp_addthreshold dbNN, "default", 200, sp_dataspace
2> go
Adding threshold for segment 'default' at '200' pages.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
(return status = 0)
```

11. You have a table assigned to a specific segment. You want to report when that table reaches 25%, 50%, 65%, and 80% capacity of the segment. This reporting should be done automatically. The total number of available pages in the segment is 1000.

a. What do you do (list the commands, also)?

---

1. Check that the threshold settings do not violate the minimum interval between thresholds. The minimum interval between thresholds is 2 * @@thresh_hysteresis. This value is fixed, and can be found as follows:

```
1> select @@thresh_hysteresis
2> go
-----------
         64

(1 row affected)
```

Thresholds must be at least 128 pages from each other. For this scenario, the thresholds would need to be set up at 750 (25% used), 500 (50% used), 350 (65% used) and 200 (80% used), so the 128 page minimum separation is achievable.

2. Create a threshold procedure for the local database.

3. Add the threshold using sp_addthreshold. The command sequence would be:

```
1> sp_addthreshold dept42, "default", 750, space_monitor
2> go
Adding threshold for segment 'default' at '750' pages.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
(return status = 0)
1> sp_addthreshold dept42, "default", 500, space_monitor
```

```
2> go
Adding threshold for segment 'default' at '500' pages.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
(return status = 0)
1> sp_addthreshold dept42, "default", 350, space_monitor
2> go
Adding threshold for segment 'default' at '350' pages.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
(return status = 0)
1> sp_addthreshold dept42, "default", 200, space_monitor
2> go
Adding threshold for segment 'default' at '200' pages.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
(return status = 0)
```

4. Test the thresholds.

You can test the thresholds in a variety of ways, all of which involve generating a substantial numbers of data records. There is a potential problem with the log filling up, if this is not handled by an existing LCT (Last Chance Threshold); for test purposes, the database option "trunc log on chkpt" could be set. Here's an example:

```
1> use master
2> go
1> sp_dboption dept42, "trunc log on chkpt", true
2> go
Database option 'trunc log on chkpt' turned ON for database 'dept42'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use dept42
2> go
1> checkpoint
2> go
1> insert sm_table values(3,"another text string")
2> go 200000 /* use 'go NNNN' with a large number */
```

If all is working correctly, the following will appear in the errorlog:

```
00:94/02/16 09:14:35.97 server  background task message: THRESHOLD
WARNING: database 'dept42', segment 'default' at '500' pages
00:94/02/16 09:18:32.75 server  background task message: THRESHOLD
WARNING: database 'dept42', segment 'default' at '350' pages
00:94/02/16 09:21:15.41 server  background task message: THRESHOLD
WARNING: database 'dept42', segment 'default' at '200' pages
```

b. Write a simple free space threshold procedure named space_monitor that prints a message to the error log indicating the number of pages left. For example:

```
create procedure space_monitor
    @dbname     varchar(30),
    @segmentname    varchar(30),
    @space_left int,
    @status     int
as
begin
    print "THRESHOLD WARNING: database '%1!', segment '%2!' at '%3!' pages",
    @dbname, @segmentname, @space_left
end
```

FINDING ROOT CAUSES AND PROPOSING SOLUTIONS

12. A user calls and tells you that he/she has executed a transaction but the system just hangs, not returning control to the client application.

a. Before looking at the error log, what are two possible reasons this might happen?

---

1. Last change threshold stored procedure is not defined.
2. The threshold procedure is in the middle of a dump transaction and you are suspended.
3. There are conflicting locks.

b. There are actually conflicting locks on the table this user is attempting to access. How do you find who is causing you to hang and the name of the object in the database that is being locked out?

---

1. Run sp_who to determine who is blocking who. For example:

```
1> sp_who
2> go
 spid   status        loginame     hostname   blk   dbname     cmd
 ------ ------------ ------------ ---------- ----- ---------- ----------------
      1 recv sleep   sa           beast      0     dept42     AWAITING COMMAND
      2 sleeping     NULL                    0     master     NETWORK HANDLER
      3 sleeping     NULL                    0     master     MIRROR HANDLER
      4 sleeping     NULL                    0     master     CHECKPOINT SLEEP
      5 running      sa           beast      0     dept42     SELECT
      6 lock sleep   sa           beast      1     dept42     UPDATE

(6 rows affected, return status = 0)
```

This shows that spid 6 is being blocked by spid 1.

2. Examine the locks that spid 1 is holding using sp_lock. For example:

```
1> sp_lock 1
2> go
The class column will display the cursor name for locks associated with a
cursor
for the current user and the cursor id for other users.
 spid   locktype                    table_id    page        dbname
        class
 ------ --------------------------- ----------- ----------- ---------------
        ------------------------------
      1 Ex_intent-blk                 128003487           0 dept42
        Non Cursor Lock
      1 Ex_page                       128003487         321 dept42
        Non Cursor Lock

(2 rows affected, return status = 0)
```

3. Determine the table name using the object_name function in the correct database. For example:

```
1> use dept42
2> go
1> select object_name(128003487)
2> go

 ------------------------------
 sm_table

(1 row affected)
```

13. You receive the following error after trying to log into server as sa:

```
% isql -Usa -P -SMARION10
Operating-system error:
      No such file or directory
DB-LIBRARY error:
      Could not open interface file.
%
```

What's the problem? How would you find the problem and then fix it?

This indicates that the interfaces file is not being found where expected, usually as a result of the SYBASE environment variable (or its equivalent) not being correctly set. Here's a Unix example:

```
% echo $SYBASE
/curdev1/server10ga/install
% setenv SYBASE /curdev1/server10ga
% isql -Usa -P -SMARION10
1>
```

AUDITING

14. A particularly sensitive table containing confidential information needs to keep track of user activity and provide the data for reporting this activity. You've attempted to create triggers that would fire and report the information, however, the amount of coding and testing necessary exceeds the time you can allot to create the application's code. You will have to install auditing. How will you plan for use of sybsecurity? What steps will you take and commands will you use to implement this plan? The following information regarding the auditing of the database is defined:

- **a. 100 writes per day, 5 days per week on the audit system.**
- **b. An audit record can vary between a minimum of 20 to a maximum of 420 characters.**
- **c. Audit system is backed up, then purged every three months (13 weeks in a quarter). There is about a one-week overlap from quarter to quarter.**
- **d. Audit system is not mirrored.**
- **e. Audit system is maintained on separate device.**
- **f. You will audit inserts, updates, deletes, and selects on the table in your db.**

How will you plan for use of sybsecurity? What steps will you take and commands will you use to implement this plan?

---

1. Calculate the space required and proceed to install sybsecurity with sybinit with an appropriately sized device in place.

100 writes per day, 5 days per week, for 3 months is 7000 records, assuming a 1 week overlap. An audit record can vary between a minimum of 20, and a maximum of 420 characters; thus anywhere between 4 and 90 records can fit on a single 2k page. The size of sysaudits for this amount of data could vary between 80 pages (160kb) and 1750 pages (approximately 3.5Mb). To allow leeway for future growth, it would be prudent to install sybsecurity specifying at least 10Mb for the database size, with an appropriately sized device

2. Use sp_auditobject as follows to set up the required actions to be audited. For this scenario, assume that the table to be audited is sm_table, in the dept42 database. For example:

```
1> sp_auditobject sm_table, dept42, "both"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

Note that because insert, update, delete and select are going to be audited, none need to be specified as parameters to sp_auditobject.

3. Turn on auditing using sp_auditoption:

```
1> sp_auditoption "enable auditing", "on"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

4. Test the audit option. For example:

```
1> use dept42
2> go
1> insert sm_table values(1,"audit")
2> go 100
(1 row affected)
100 xacts:
1>  use sybsecurity
2> go
1> select count(*) from sysaudits
2> go

 -----------
         101

(1 row affected)
1> sp_spaceused sysaudits
2> go
 name                 rowtotal    reserved   data       index_size unused
 -------------------- ----------- ---------- ---------- ---------- ----------
 sysaudits            101         16 KB      8 KB       0 KB       8 KB

(return status = 0)
```

There are 101 rows, and 4 data pages; thus around 25 rows per page are being written. This validates the original size estimate.

LOGINS, ROLES, AND PERMISSIONS

15. The following requirements for your new database installation have just been given to you. Write out the appropriate commands and then implement them on the new Server.

a. User harry42 reports that he has forgotten his password. What steps would you take to provide him with a new one?

Login with SSO role then change harry's password using sp_password. For example:

```
1> sp_password student42, newharry, harry42
2> go
Password correctly set.
(return status = 0)
```

b. New employee Marie needs access to the SQL Server. Her login will be'marie42', with 'friday' as her password. Her default database is to be master. She will be responsible for database backups and restores, so will require the appropriate role. Outline the steps necessary to enable her to perform her tasks, then verify that the steps were performed.

---

1. Add her login, password, and default database using sp_addlogin.
2. Add her as a user using sp_adduser.
3. Grant her login OPER role using sp_role. Note: Since this is an OPER only account then having Master as the default database makes sense. If Marie has other work on the SQL Server databases to perform, then make a user database her default.
4. Verify her login and role using sp_displaylogin.

For example, the sequence of commands would be:

```
1> sp_addlogin marie42, friday, master
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
1> sp_adduser marie42
2> go
New user added.
(return status = 0)
1> sp_role "grant", "oper_role", marie42
2> go
(return status = 0)
1> sp_displaylogin marie42
2> go
Suid: 8
Loginame: marie42
Fullname:
Configured Authorization: oper_role
Locked: NO
Date of Last Password Change: Feb 18 1994  1:15PM
(return status = 0)
```

c. Now that Marie has taken over backup responsibilities, tom42 no longer requires the 'OPER' role. How would you make this change?

Revoke OPER role from tom42 using sp_role. For example:

```
1> sp_role "revoke", "oper_role", tom42
2> go
Authorization updated.
(return status = 0)
```

d. tom42, in turn, will be heading up a new project for which he will need to create and administer databases, including logins. What changes need to be made to his account to enable him to perform these tasks?

Grant tom42 create database permission in the context of tom42's default database. For example:

```
1> use dept42
2> go
1> grant create database to tom42
2> go
```

Lab 13a - Logical Database Rebuild

1. Write a small "create table" script that creates a table in your database that is structured like *pubs2..publishers*. Test it out, then drop the table you just created.

```
/* create table script */

use db42
go

create table publishers42
        (pub_id char(4),
        pub_name varchar(20),
        city varchar(20),
        state char(20))
go
```

To execute:

```
godzilla% isql -Ustudent42 -Pstudent42 < create.table
```

Drop the table:

```
godzilla% isql -Ustudent42 -Pstudent42
1> use db42
2> go
1> drop table publishers42
2> go
```

2. Use `defncopy` to copy out *pub_idrule*. Add the resulting code to your script. Run your new script, and see if the table and rule are created properly.

Copy definition out to a file:

```
godzilla% defncopy -Ustudent42 -Pstudent42 out pub.rule pubs2
pub_idrule
```

Display contents of script created by defncopy:

```
godzilla% more pub.rule
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622", "1756")
or @pub_id like "99[0-9][0-9]"
/* ### DEFNCOPY: END OF DEFINITION */
```

Add to script.

```
/* create table script */

use db42
go

create table publishers42
        (pub_id char(4),
        pub_name varchar(20),
        city varchar(20),
        state char(20))
go
```

```
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622", "1756")
or @pub_id like "99[0-9][0-9]"
go
```

Execute:

```
godzilla% isql -Ustudent42 -Pstudent42 < create.table
```

Examine table structure:

```
godzilla% isql -Ustudent42 -Pstudent42
1> use db42
2> go
1> sp_help publishers42
2> go
 Name                                Owner
         Type
 ----------------------------- ------------------------------
        ----------------------
 publishers42                        dbo
         user table

 Data_located_on_segment             When_created
 ----------------------------- -------------------------
 default                                Nov  4 1993  9:34AM

 Column_name     Type             Length Prec Scale Nulls
Default_name
         Rule_name       Identity
 --------------- --------------- ------ ---- ----- -----
---------------
        ---------------- --------
 pub_id          char                  4 NULL  NULL     0 NULL
         NULL                    0
 pub_name        varchar              20 NULL  NULL     0 NULL
         NULL                    0
 city            varchar              20 NULL  NULL     0 NULL
         NULL                    0
 state           char                 20 NULL  NULL     0 NULL
         NULL                    0
Object does not have any indexes.
No defined keys for this object.

(return status = 0)
```

Examine rule:

```
1> sp_helptext pub_idrule
2> go
 # Lines of Text
 ---------------
               1

(1 row affected)

         text
```

```
------------------------------------------------------------------
------
------------------------------------------------------------------
--------------
------------------------------------------------------------------
--------------
-----------------------


create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622", "1756")
or @pub_id like "99[0-9][0-9]"


(1 row affected, return status = 0)
```

3. Load your new table with data from *pubs2..publishers* using bulk copy commands shown in this module. Check to see that the data was loaded.

To copy rows from pubs2..publishers out to a file:

```
godzilla% bcp pubs2..publishers out publishers.dat -c -Ustudent42
-Pstudent42

Starting copy...

3 rows copied.
Clock Time (ms.): total = 67     Avg = 22     (44.78 rows per sec.)
```

To load the data into db42..publishers42:

```
godzilla% bcp db42..publishers42 in publishers.dat -c -Ustudent42
-Pstudent42

Starting copy...

3 rows copied.
Clock Time (ms.): total = 69     Avg = 23     (43.48 rows per sec.)
```

If the "select into/bulkcopy" option is not set for your database, you will get the following error message:

```
Msg 4806, Level 16, State 1:
You cannot run the non-logged version of bulk copy in this
database.
Please check with the DBO.
```

(If there were indexes or triggers on your table, you would not get this message since a slower version of bulk copy would be used, one that does not require the "select into/bulkcopy" option.)

To set the "select into/bulkcopy" option in the database:

```
godzilla% isql -Ustudent42 -Pstudent42
1> use master
2> go
1> sp_dboption db42, "select into/bulkcopy", true
2> go
```

```
Database option 'select into/bulkcopy' turned ON for database
'db42'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use db42
2> go
1> checkpoint
2> go
```

Then use bcp to load the data:

```
godzilla% bcp db42..publishers42 in publishers.dat -c -Ustudent42
-Pstudent42

Starting copy...

3 rows copied.
Clock Time (ms.): total = 69     Avg = 23     (43.48 rows per sec.)
```

Verify that the data was loaded:

```
godzilla% isql -Ustudent42 -Pstudent42
1> use db42
2> go
1> select * from publishers42
2> go
 pub_id pub_name             city                 state
 ------ -------------------- -------------------
--------------------
 0736   New Age Books        Boston               MA
 0877   Binnet & Hardley     Washington           DC
 1389   Algodata Infosystems Berkeley             CA

(3 rows affected)
```

Lab 13b - Remote Access

1. What steps would you have to go through on your SQL Server to be able to send remote procedure calls to one of the other SQL Servers in the class? Which files or tables would be affected?

To set up Server originating the request (KINGKONG10):

First add the local and remote Server names to sysservers using sp_addserver:

```
1> sp_addserver KINGKONG10, local
2> go
Adding server 'KINGKONG10', physical name 'KINGKONG10'
Server added.
(return status = 0)
1> sp_addserver GODZILLA10
2> go
Adding server 'GODZILLA10', physical name 'GODZILLA10'
Server added.
(return status = 0)
```

Then configure for remote access:

```
1> sp_configure "remote access", 1
2> go
Configuration option changed. Run the RECONFIGURE command to
install.
(return status = 0)
1> reconfigure
2> go
```

Then reboot for this to take effect (startserver must be run by sybase):

```
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
godzilla% startserver -f RUN_KINGKONG10
```

2. What steps would you have to go through to arrange for another SQL Server to be able to execute procedures on your SQL Server? You would like all attempts from that SQL Server to be mapped to identical logins.

To set up Server receiving the request (GODZILLA10):

Add the name of the server to be originating the request:

```
1> sp_addserver KINGKONG10
2> go
Adding server 'KINGKONG10', physical name 'KINGKONG10'
Server added.
(return status = 0)
```

Configure for remote access:

```
1> sp_configure "remote access", 1
```

```
2> go
Configuration option changed. Run the RECONFIGURE command to
install.
(return status = 0)
1> reconfigure
2> go
```

Reboot:

```
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
godzilla% startserver -fRUN_GODZILLA10
```

Define mappings of remote logins (Here, we are mapping to identical logins. Make sure the login which will be sending the request exists on the server receiving the request.)

```
1> sp_addremotelogin KINGKONG10
2> go
New remote login created.
(return status = 0)
1> sp_addlogin student42, student42
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
```

Add appropriate logins, users, permissions.

Sample RPC from KINGKONG10 to GODZILLA10:

```
1> exec GODZILLA10...sp_who
2> go
 spid   status       loginame                        hostname   blk
        dbname                          cmd
 ------ ------------ ------------------------------ ----------
-----
        ------------------------------ ----------------
      1 recv sleep   sa                             godzilla   0
        pubs2                          AWAITING COMMAND
      2 sleeping     NULL                                       0
        master                         NETWORK HANDLER
      3 sleeping     NULL                                       0
        master                         MIRROR HANDLER
      4 sleeping     NULL                                       0
        sybsecurity                    AUDIT PROCESS
      5 sleeping     NULL                                       0
        master                         CHECKPOINT SLEEP
      6 running      student42                      godzilla   0
        master                         SELECT
      7 recv sleep   NULL                           KINGKONG10   0
        master                         SITE HANDLER

(7 rows affected, return status = 0)
```